

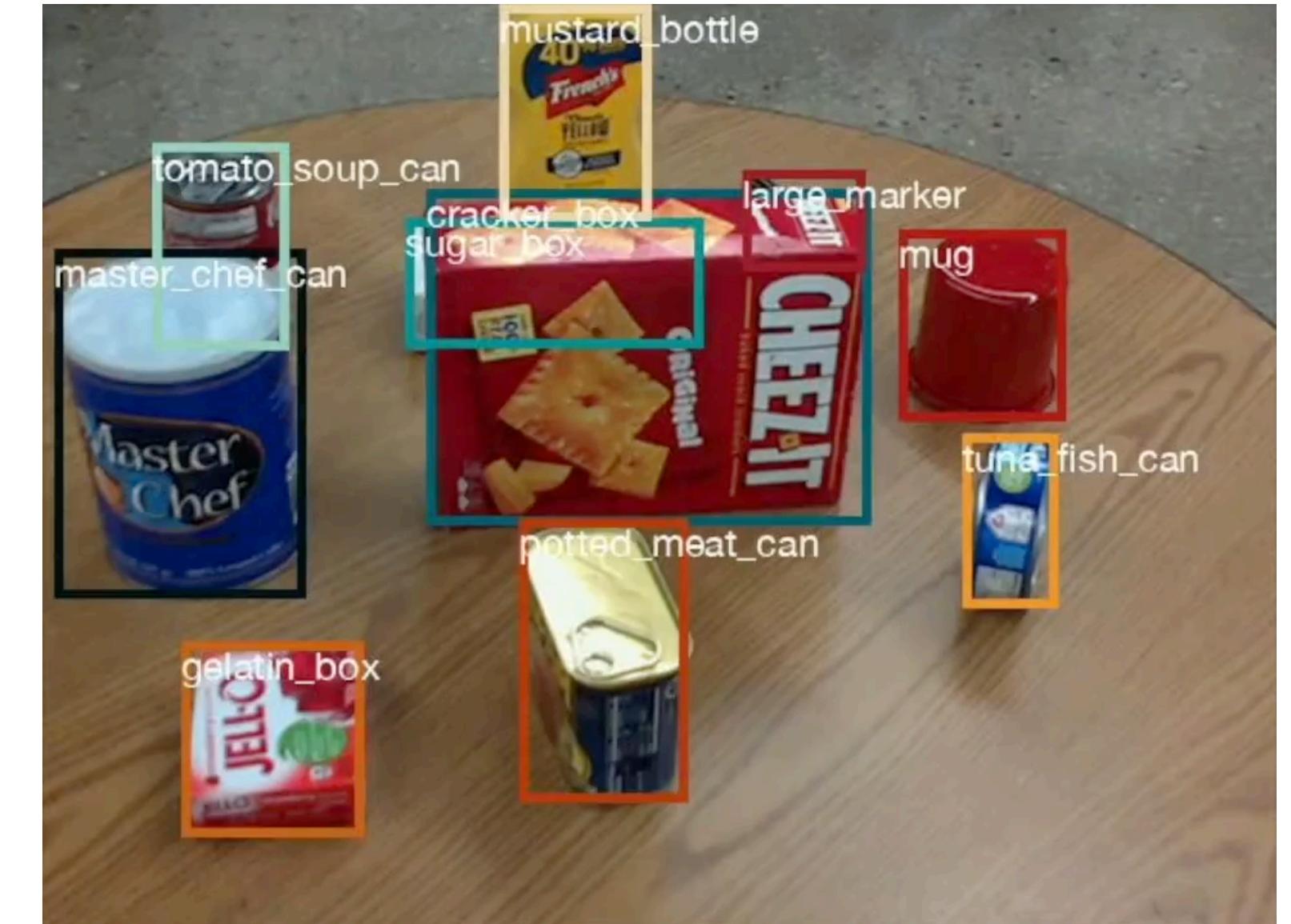
DeepRob

Lecture 12
Object Detection
University of Michigan and University of Minnesota



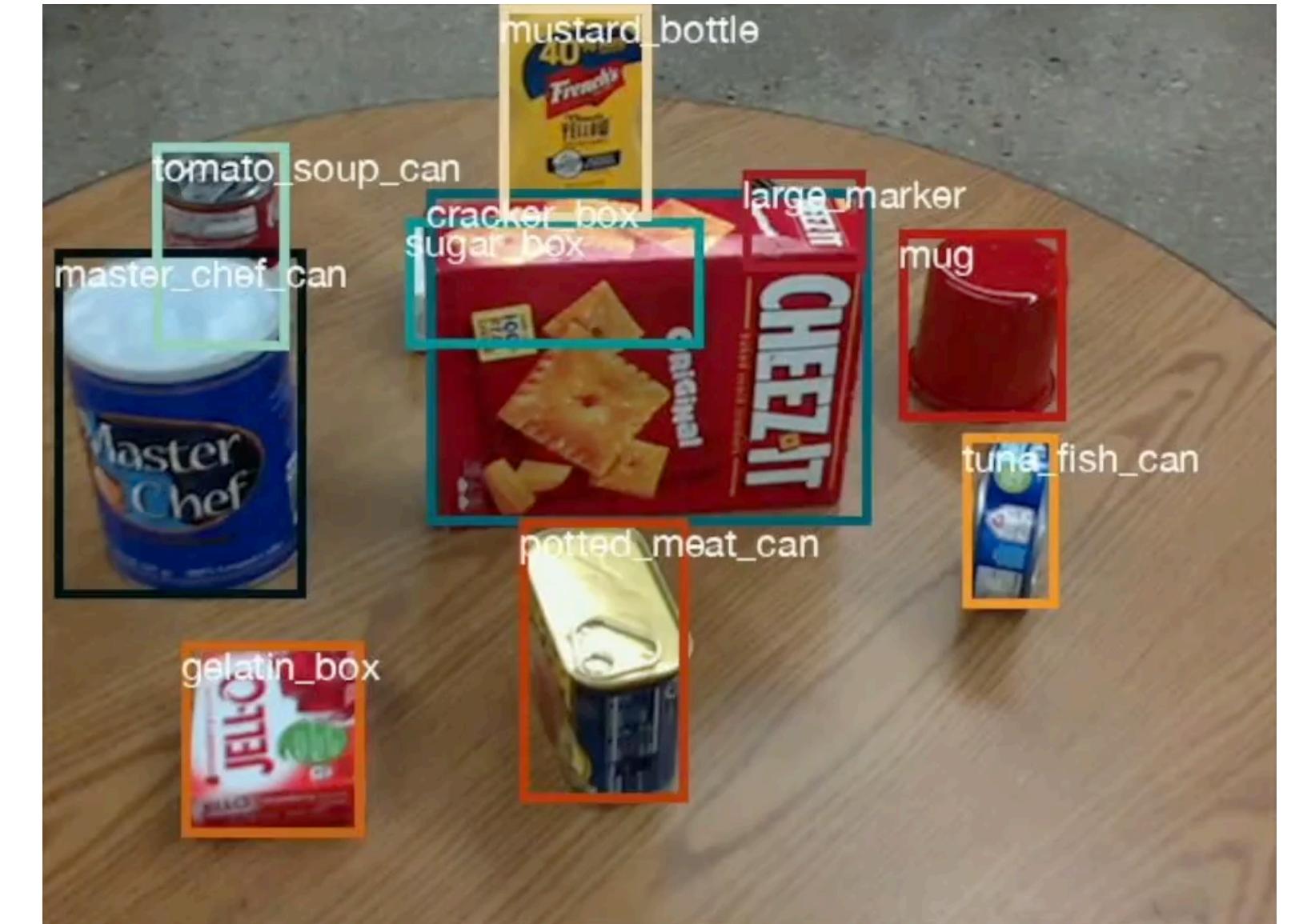
Project 3 Released

- Instructions available on the website
 - Here: deeprob.org/projects/project3/
 - New [PROPS Detection dataset](#)
- Implement CNN for classification and Faster R-CNN for detection
- Due Tuesday, February 28th 11:59 PM EST



Project 3 Released

- Instructions available on the website
 - Here: deeprob.org/projects/project3/
 - New [PROPS Detection dataset](#)
- Implement CNN for classification and Faster R-CNN for detection
- Due Tuesday, February 28th 11:59 PM EST





Recap: Deep Learning Software

Static Graphs vs Dynamic Graphs

PyTorch vs TensorFlow



So far: Image Classification

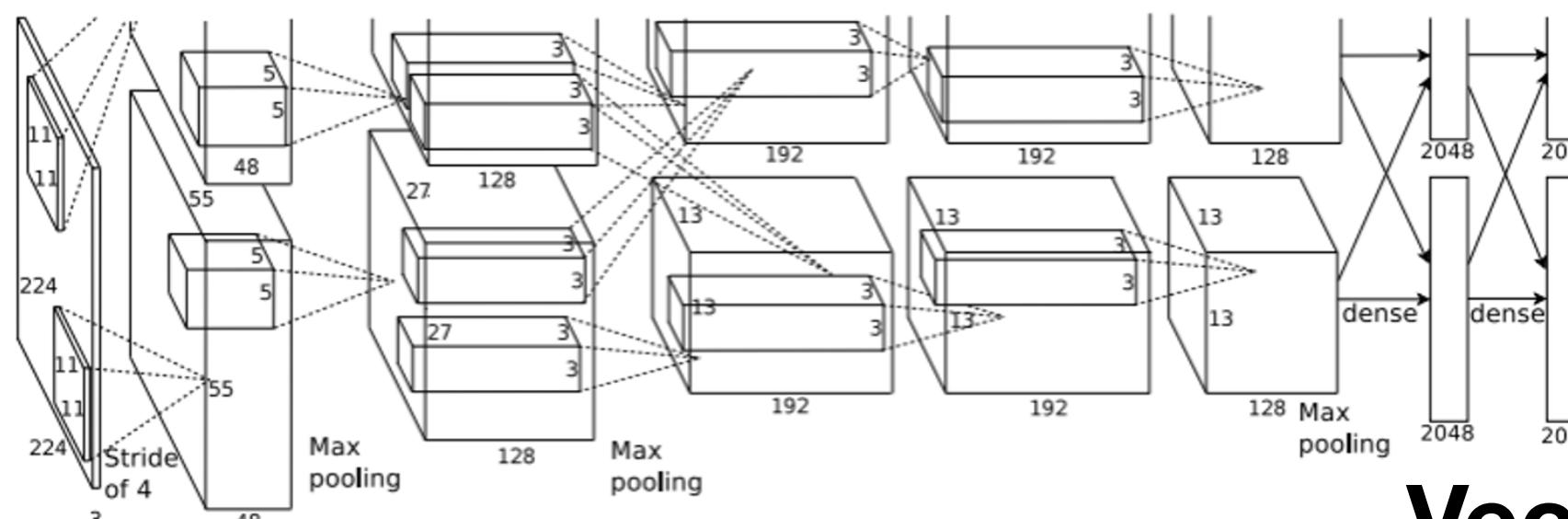


Figure copyright Alex Krizhevsky, Ilya Sutskever, and Geoffrey Hinton, 2012. Reproduced with permission.

Fully connected:

4096 to 10

Vector:

4096

Chocolate Pretzels

Granola Bar

Potato Chips

Water Bottle

Popcorn

Computer Vision Tasks

Classification



“Chocolate Pretzels”

No spatial extent



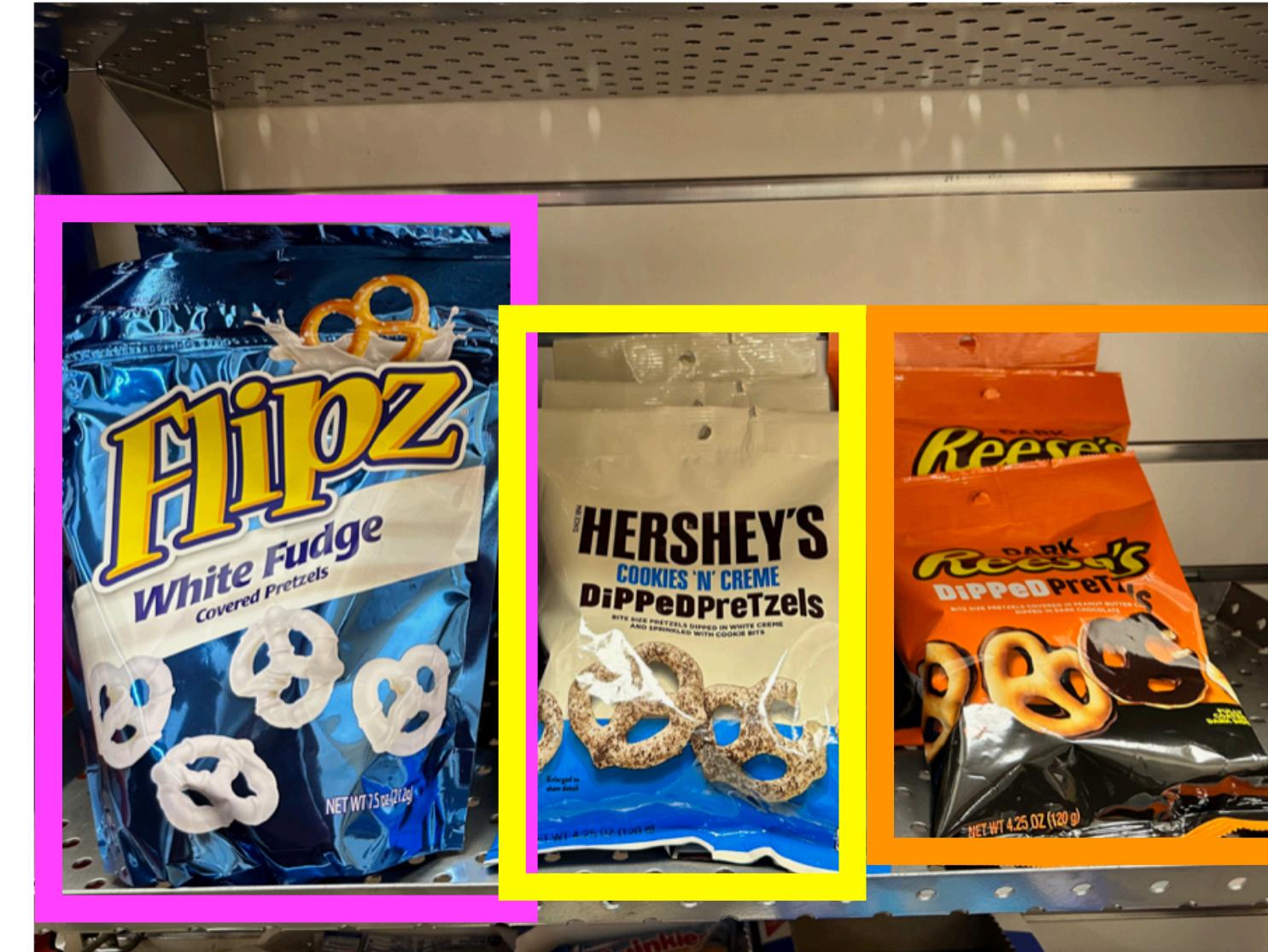
Semantic Segmentation



Chocolate Pretzels,
Shelf

No objects, just pixels

Object Detection



Flipz, Hershey's, Keese's

Multiple objects

Instance Segmentation



Computer Vision Tasks

Classification



"Chocolate Pretzels"

No spatial extent



Semantic Segmentation



Chocolate Pretzels,
Shelf

No objects, just pixels

Object Detection



Flipz, Hershey's, Keese's

Multiple objects

Instance Segmentation



Transfer Learning: Generalizing to New Tasks



Transfer Learning with CNNs

1. Train on ImageNet

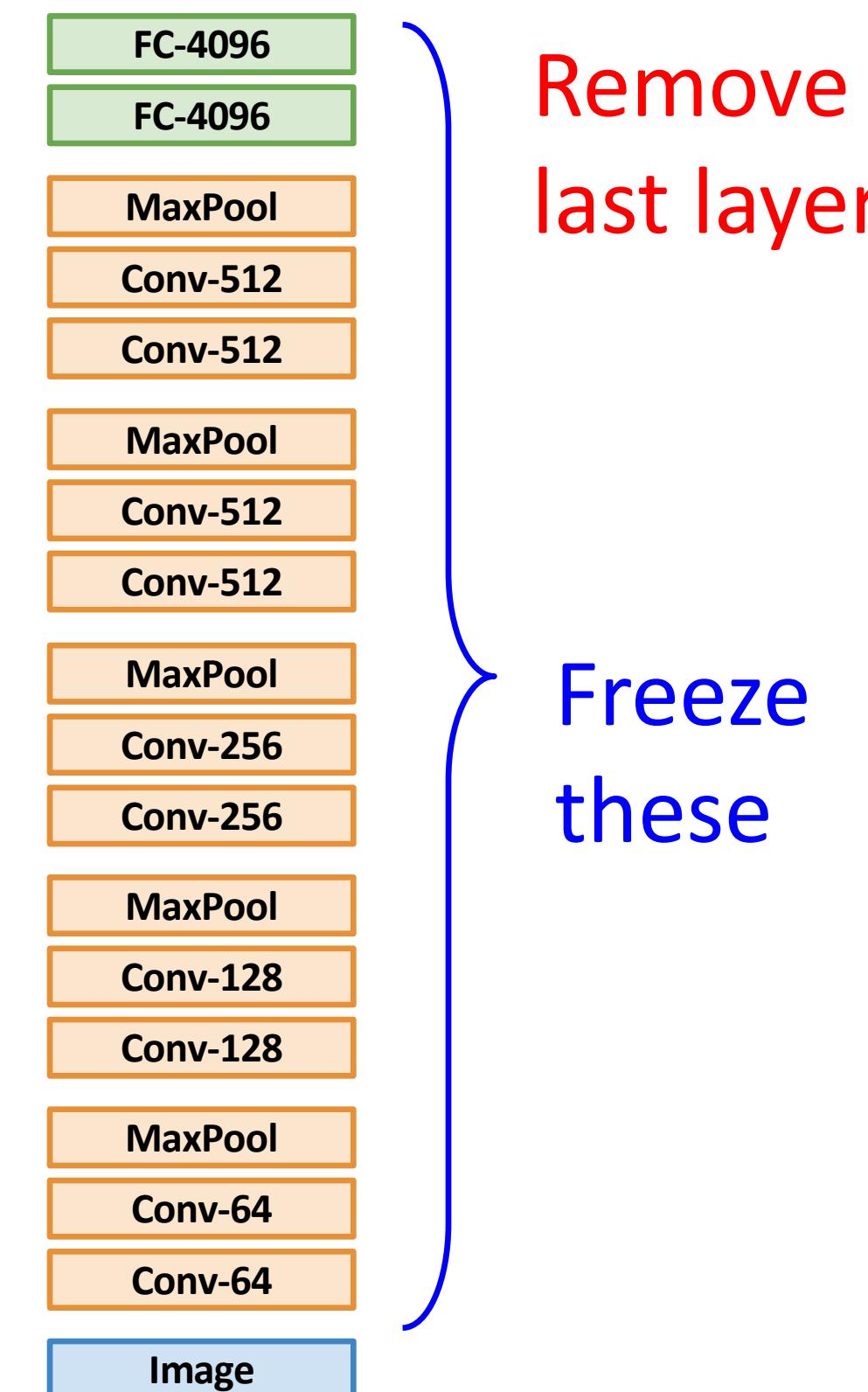


Transfer Learning with CNNs

1. Train on ImageNet



2. Use CNN as a feature extractor

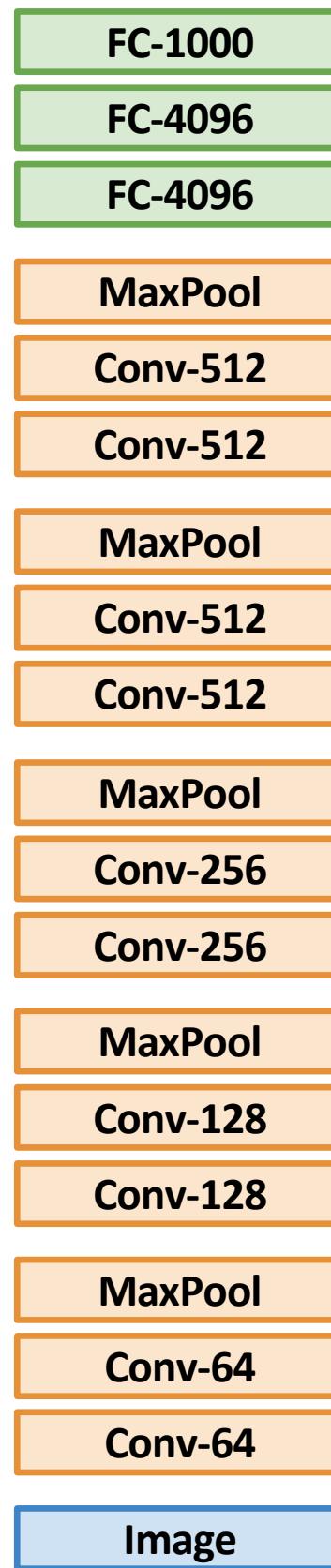


Remove
last layer

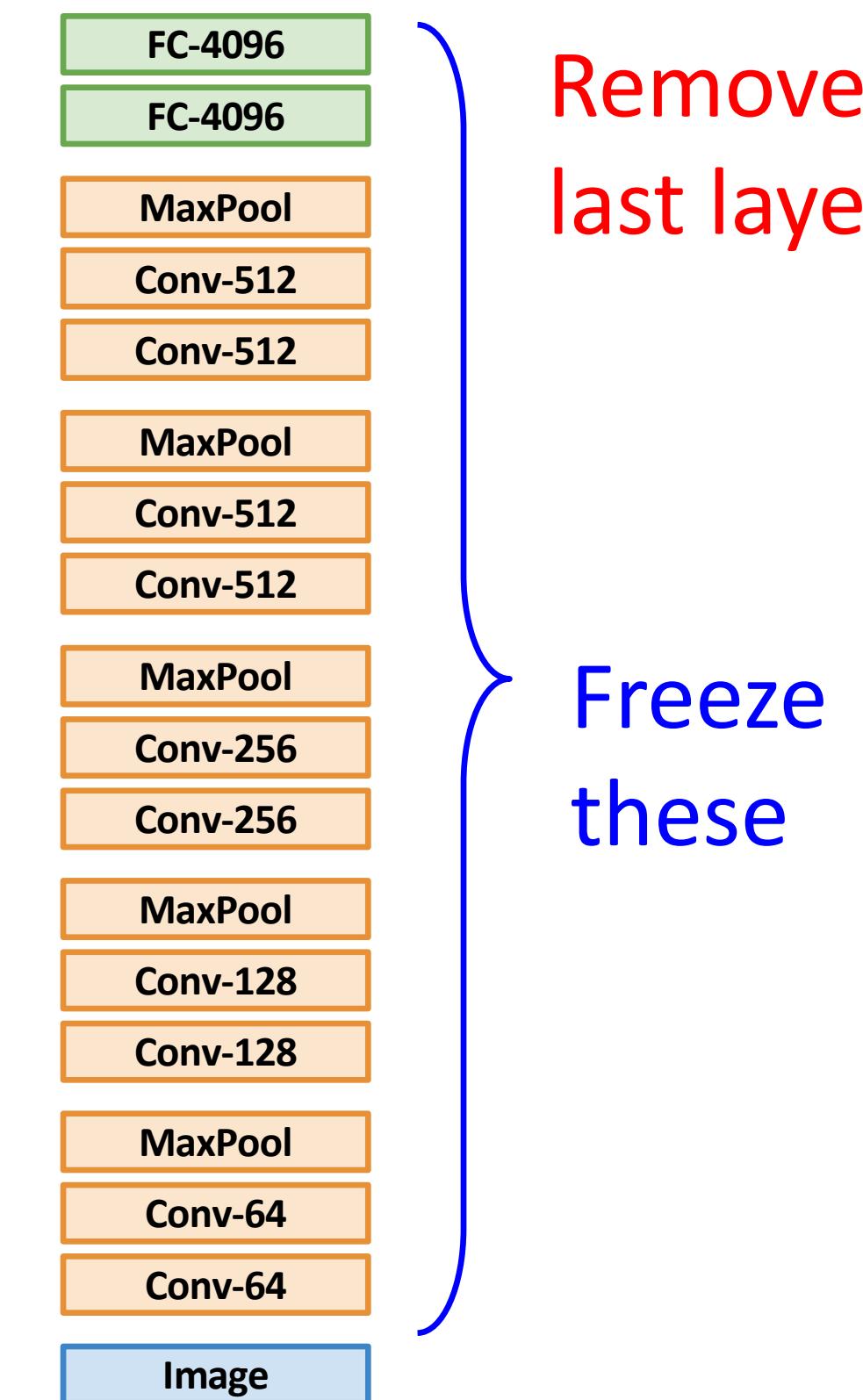
Freeze
these

Transfer Learning: Feature Extraction

1. Train on ImageNet



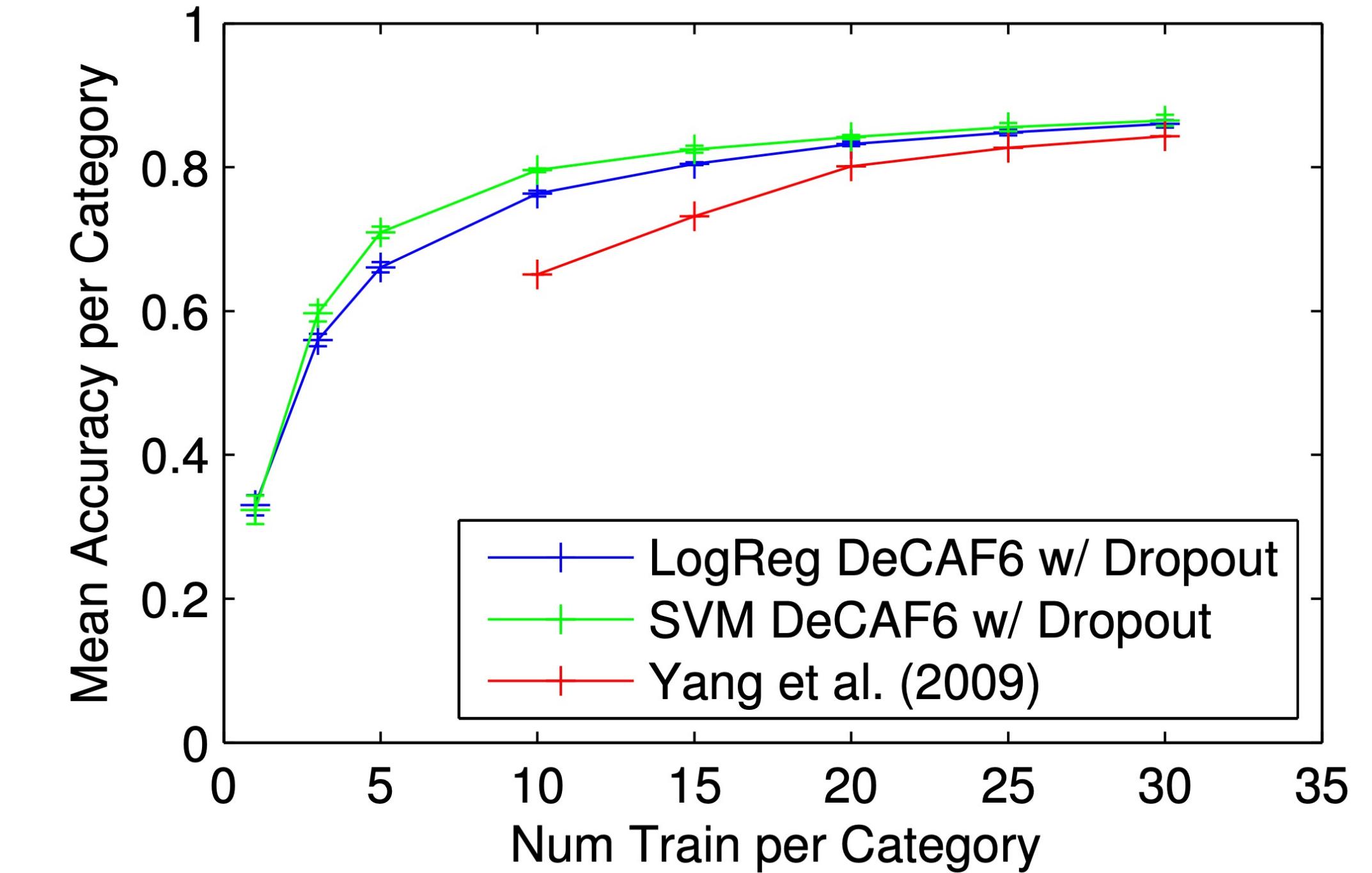
2. Use CNN as a feature extractor



Remove
last layer

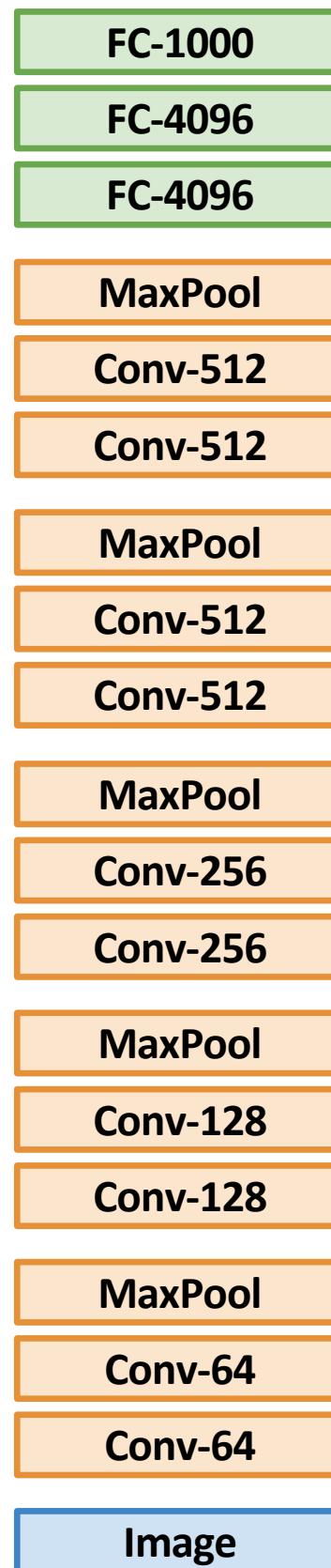
Freeze
these

Classification on Caltech-101

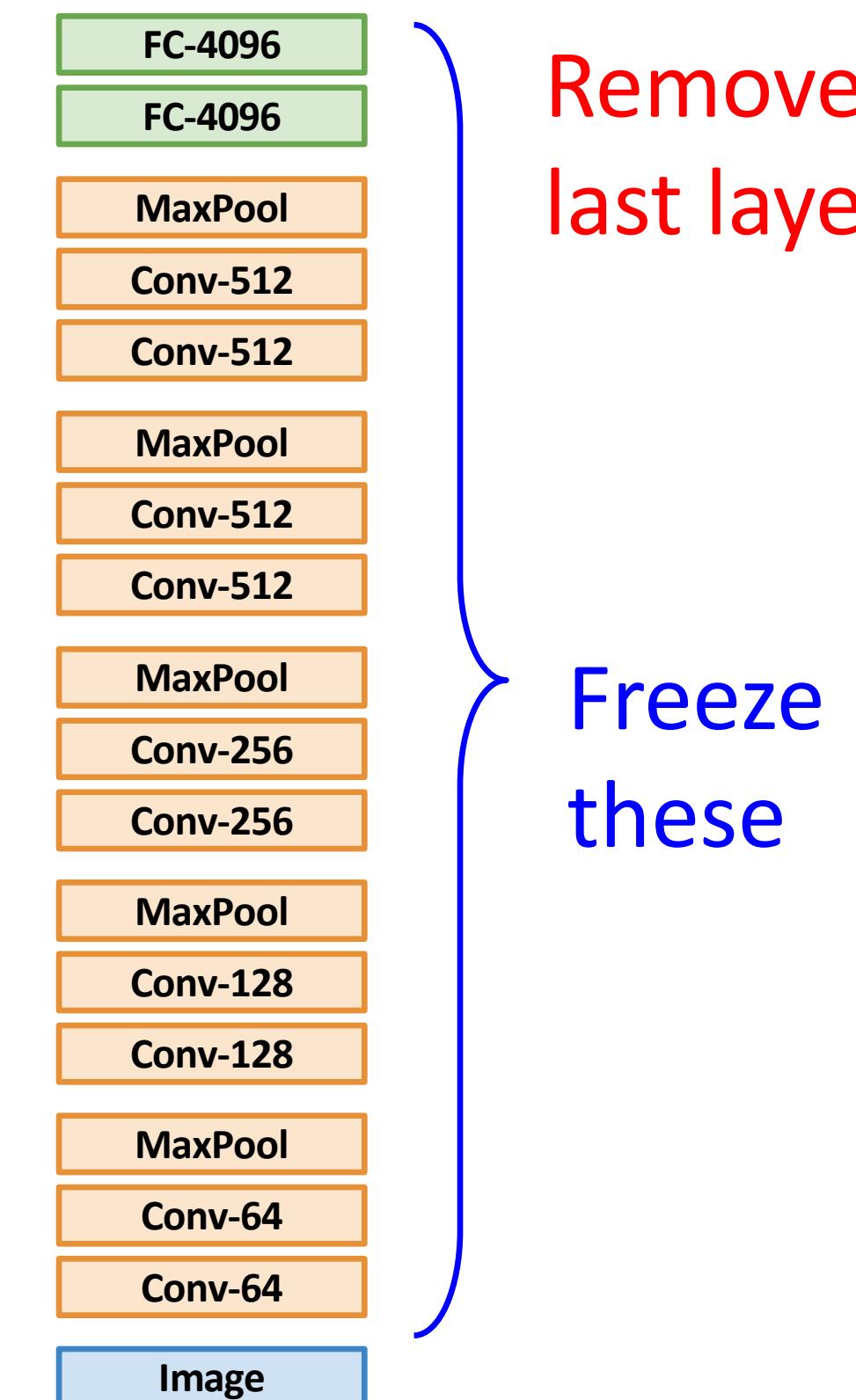


Transfer Learning: Feature Extraction

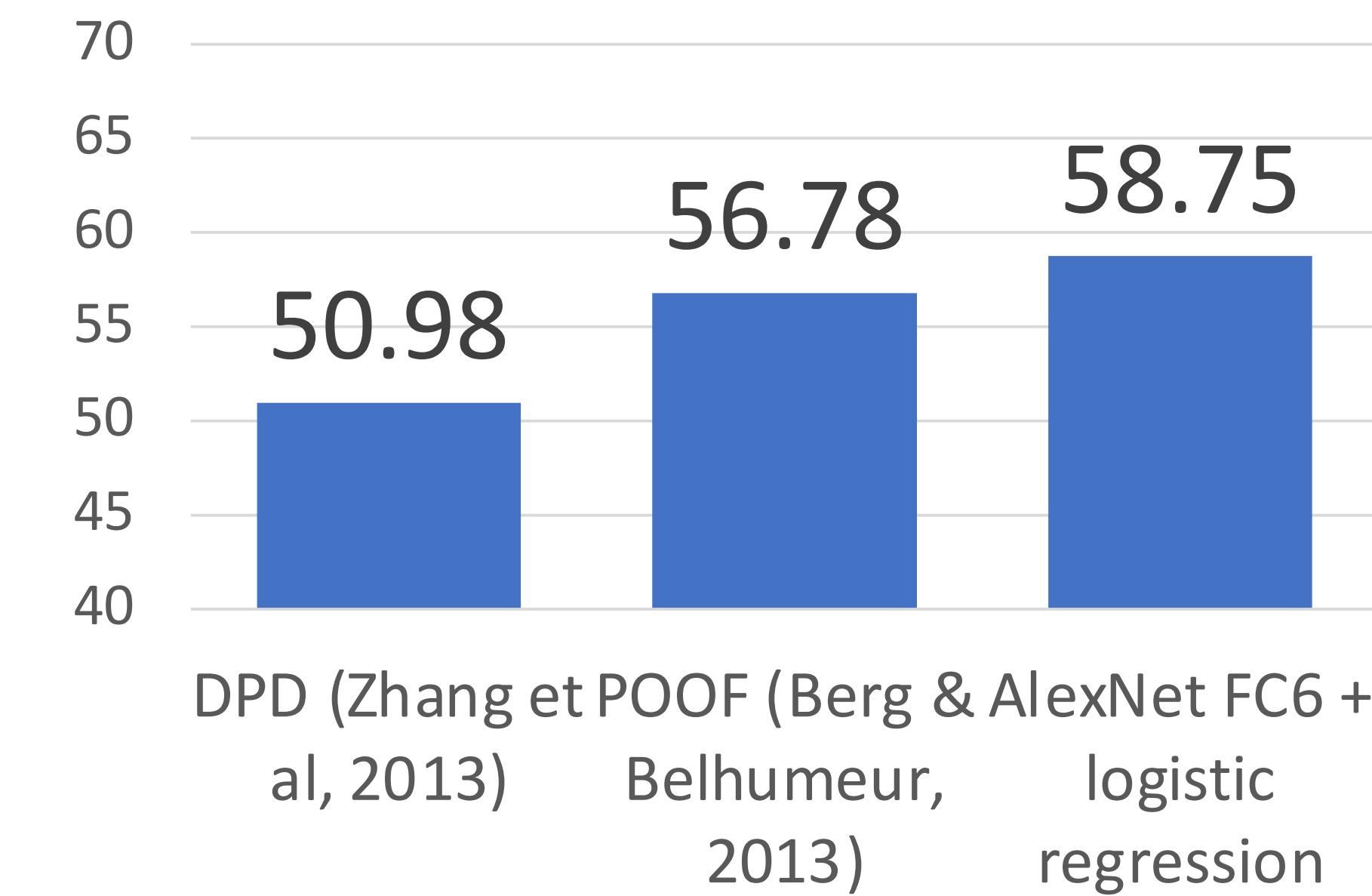
1. Train on ImageNet



2. Use CNN as a feature extractor

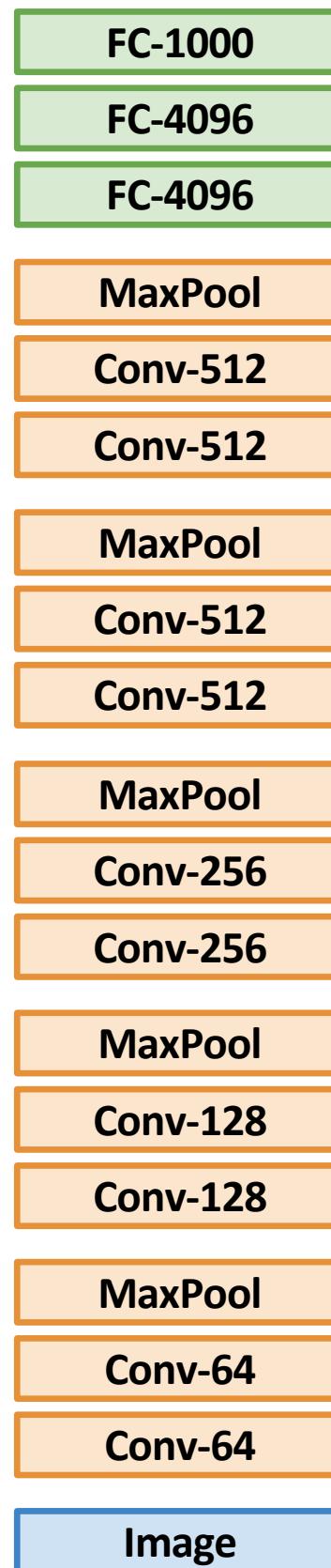


Bird Classification on Caltech-UCSD



Transfer Learning: Feature Extraction

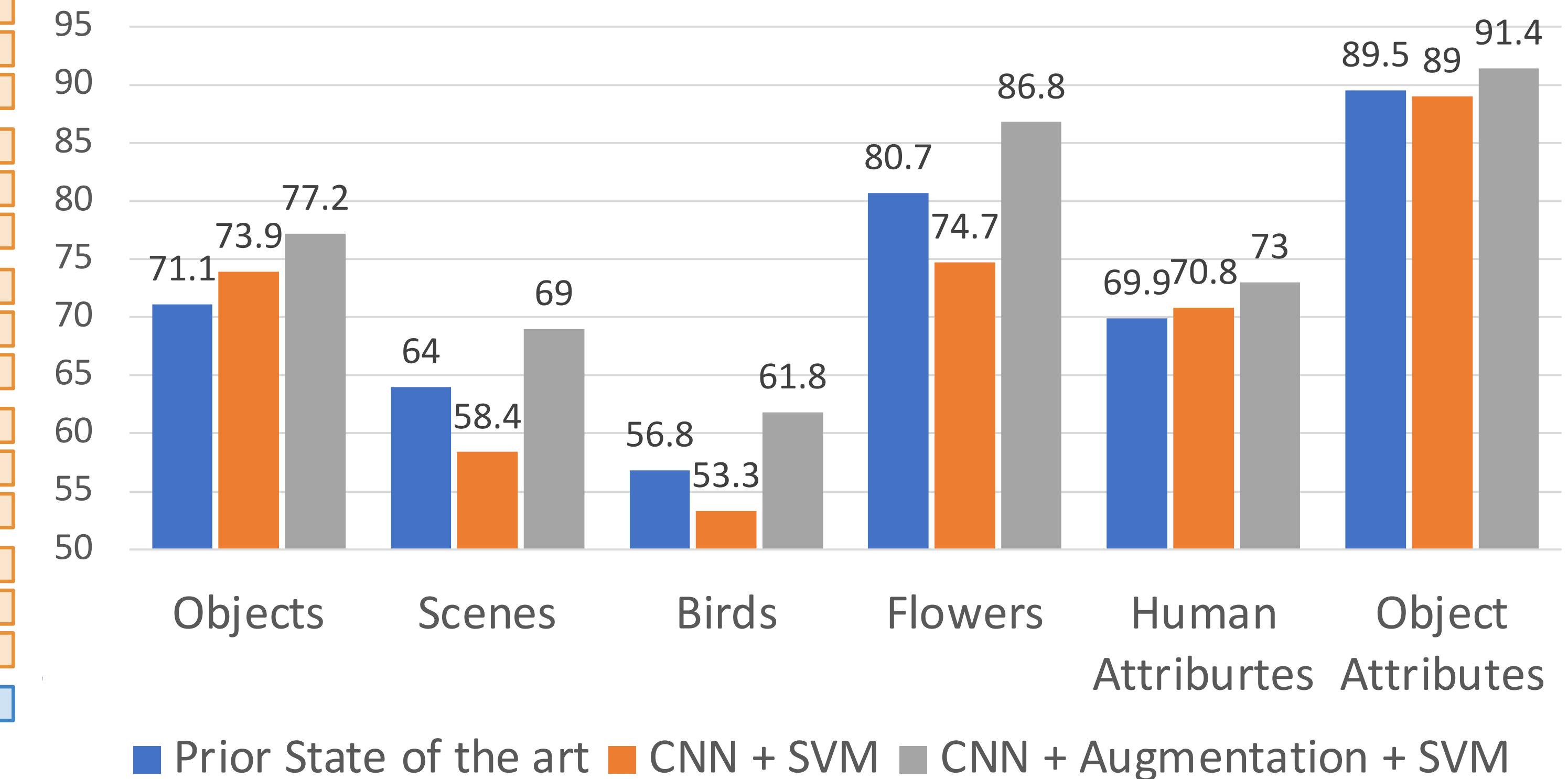
1. Train on ImageNet



2. Use CNN as a feature extractor

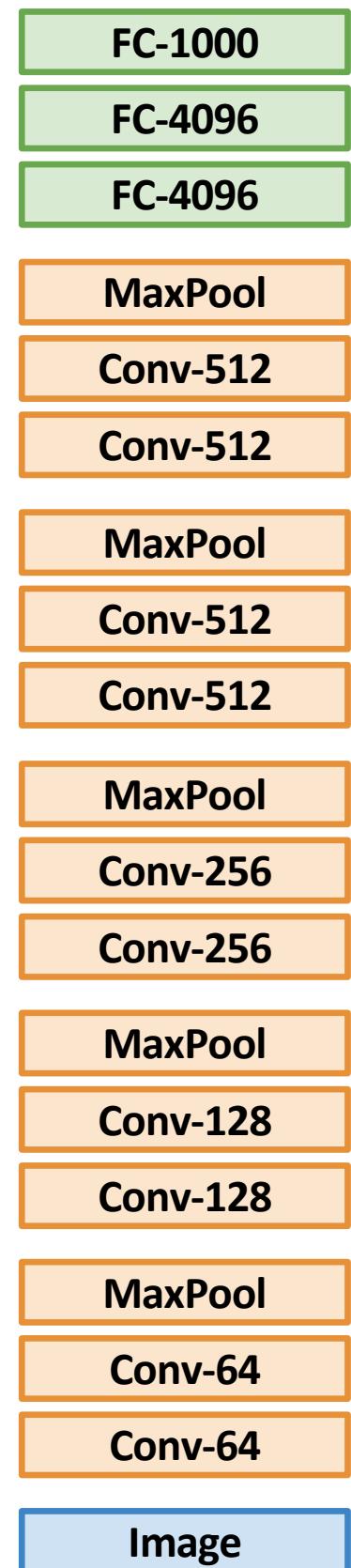


Image Classification



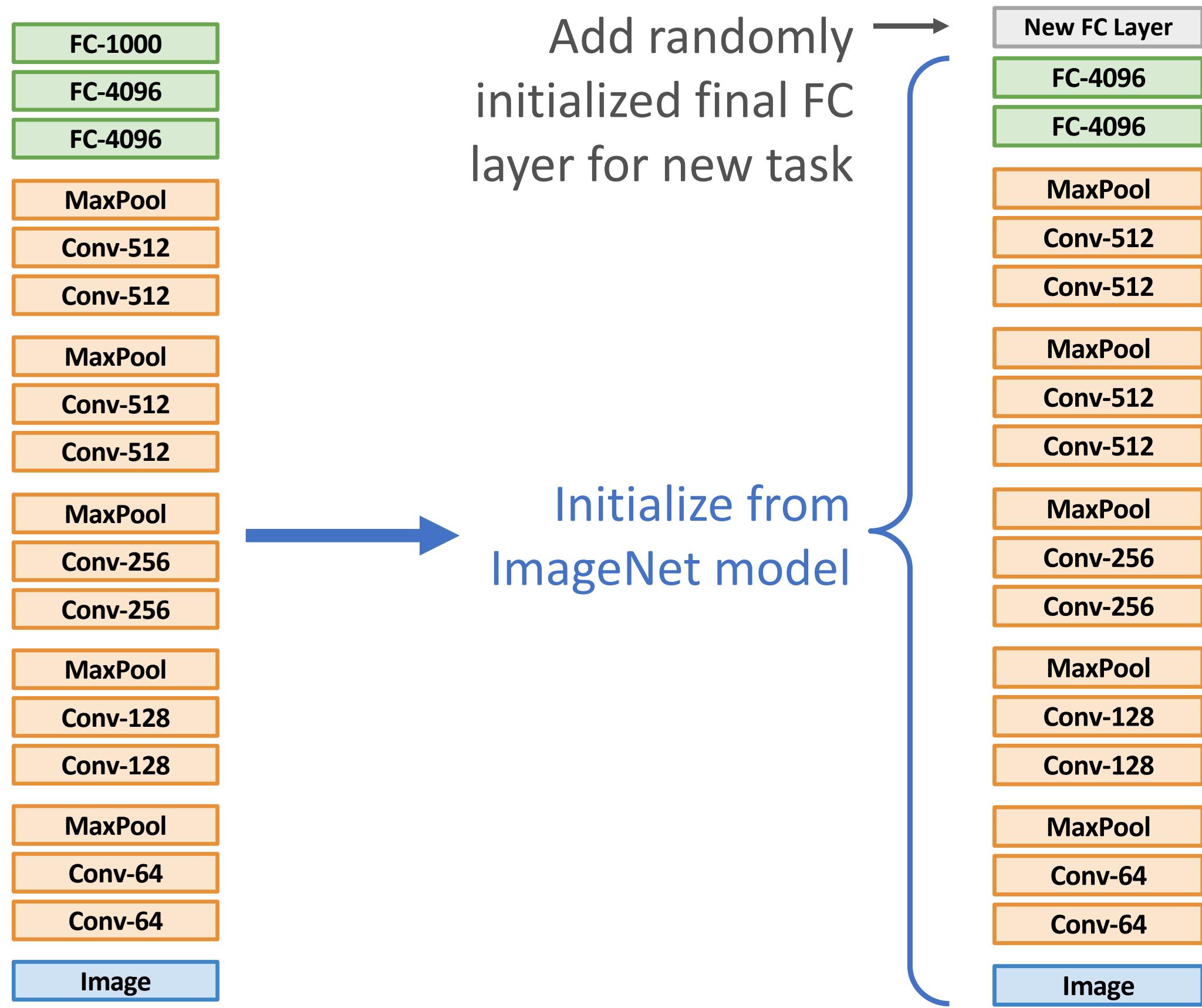
Transfer Learning: Fine Tuning

1. Train on ImageNet



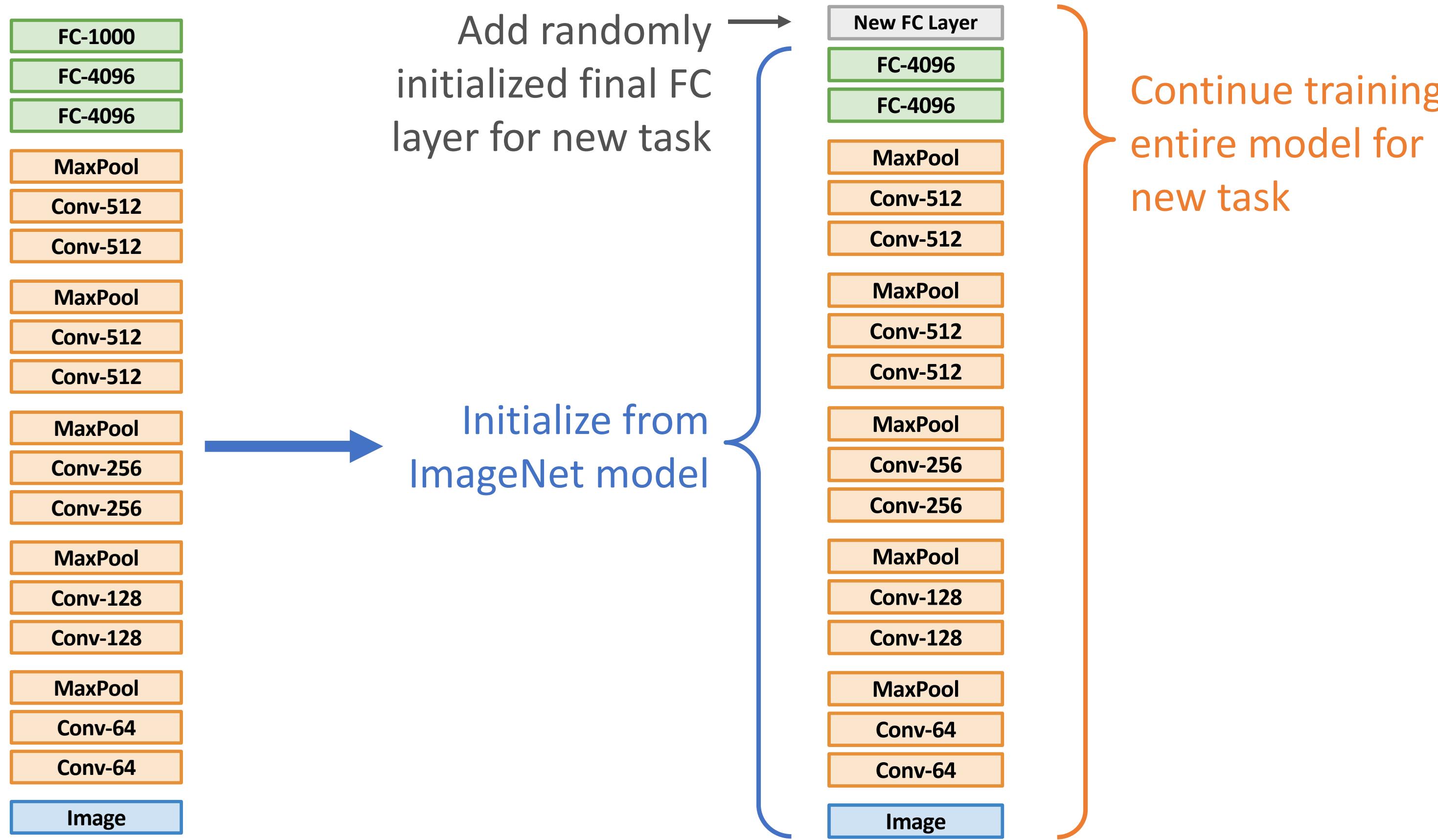
Transfer Learning: Fine Tuning

1. Train on ImageNet



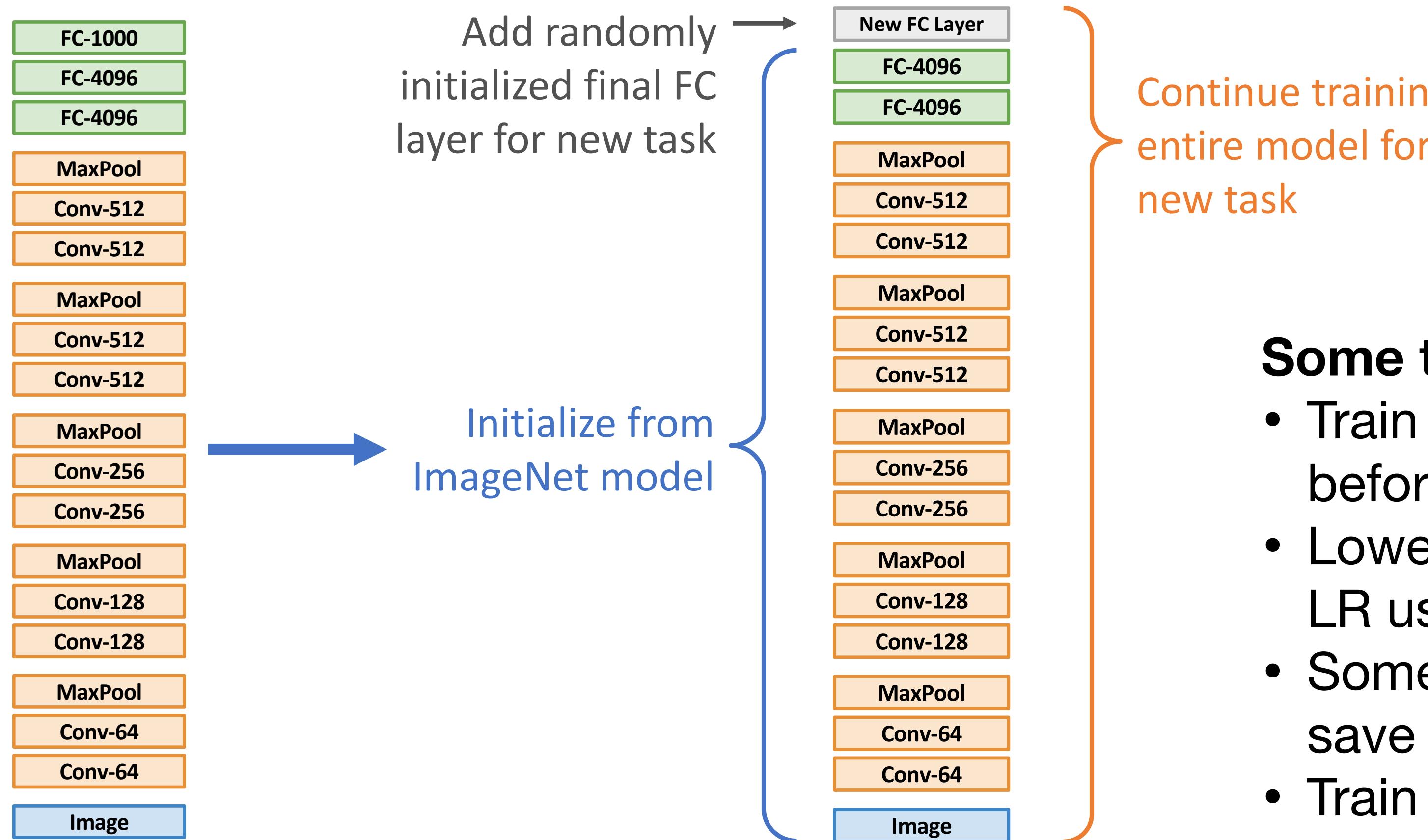
Transfer Learning: Fine Tuning

1. Train on ImageNet



Transfer Learning: Fine Tuning

1. Train on ImageNet

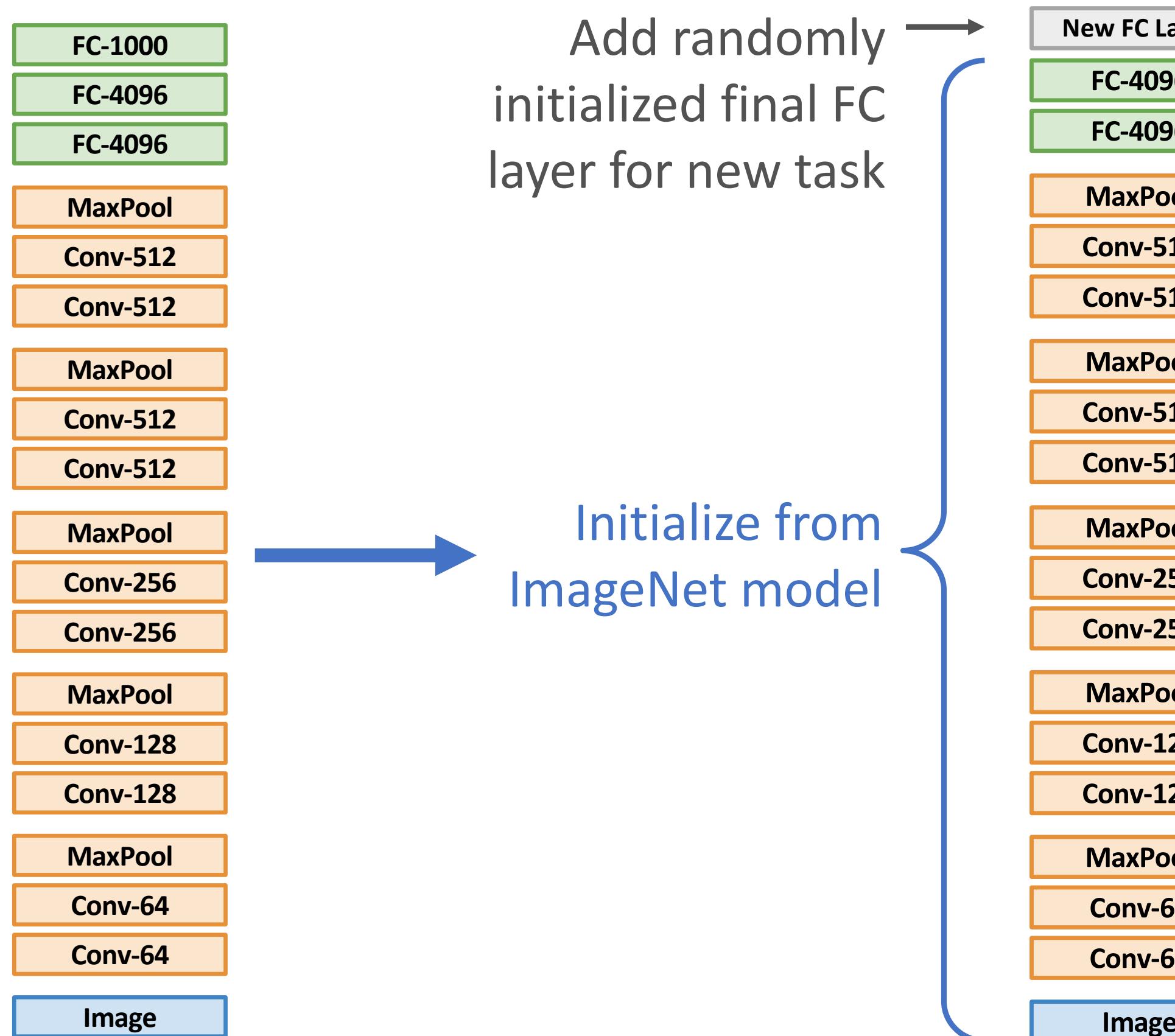


Some tricks:

- Train with feature extraction first before finetuning
- Lower the learning rate: use $\sim 1/10$ of LR used in original training
- Sometimes freeze lower layers to save computation
- Train with BatchNorm in “test” mode

Transfer Learning: Fine Tuning

1. Train on ImageNet

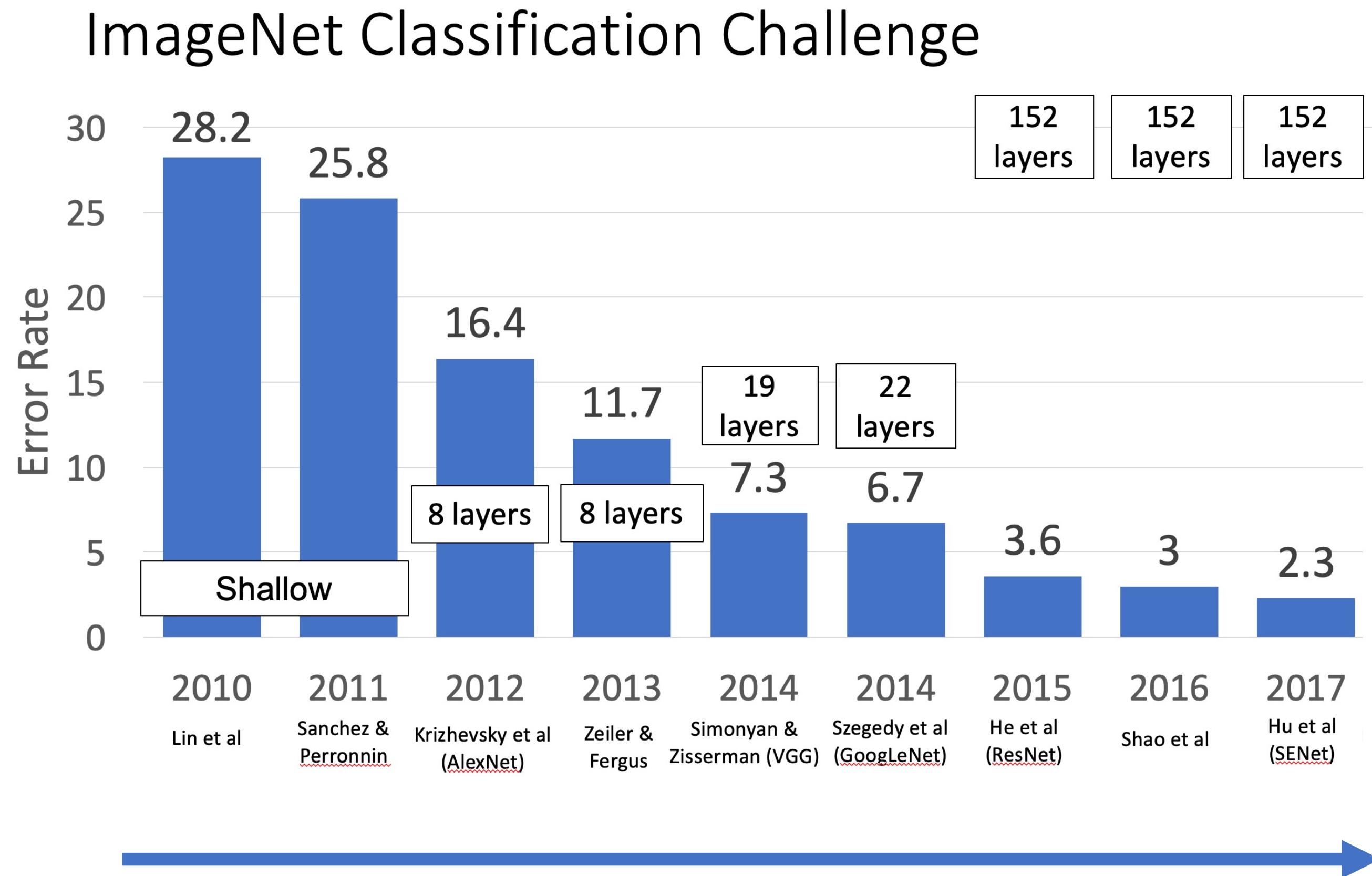


Continue training entire model for new task

Compared with feature extraction, fine-tuning:

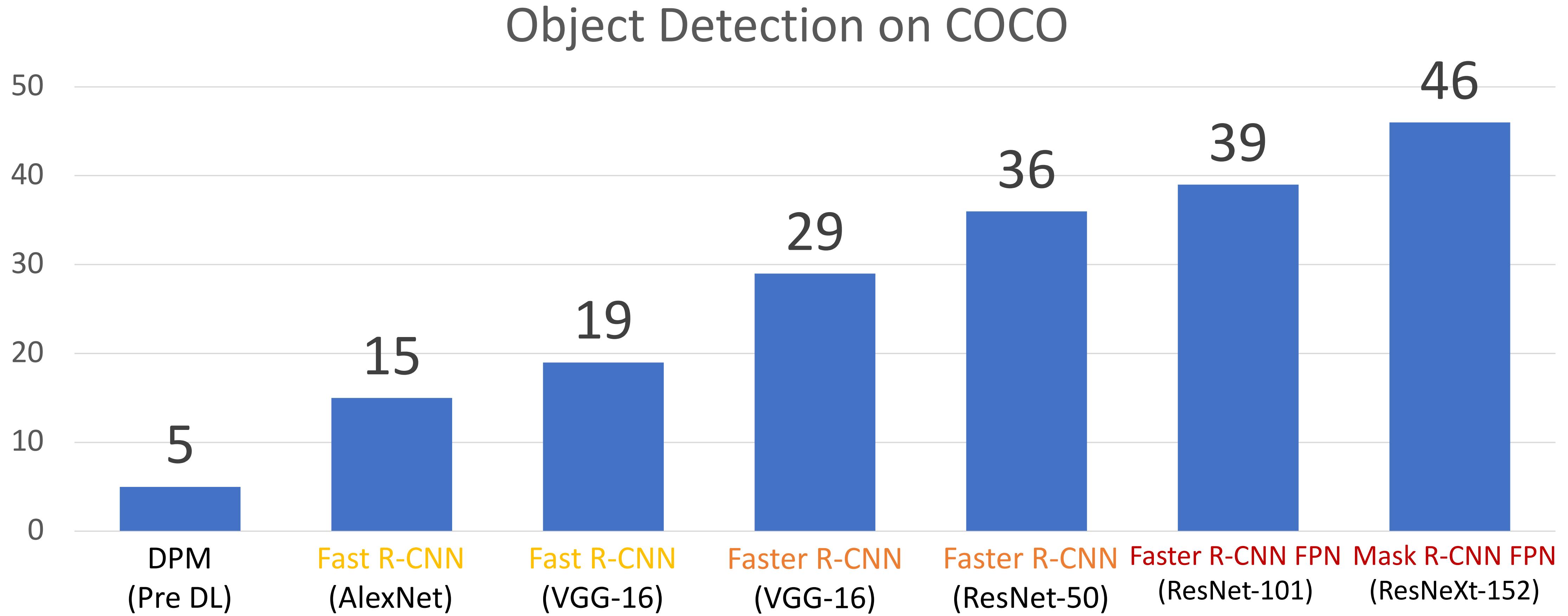
- Requires more data
- Is computationally expensive
- Can give higher accuracies

Transfer Learning: Architecture Matters!



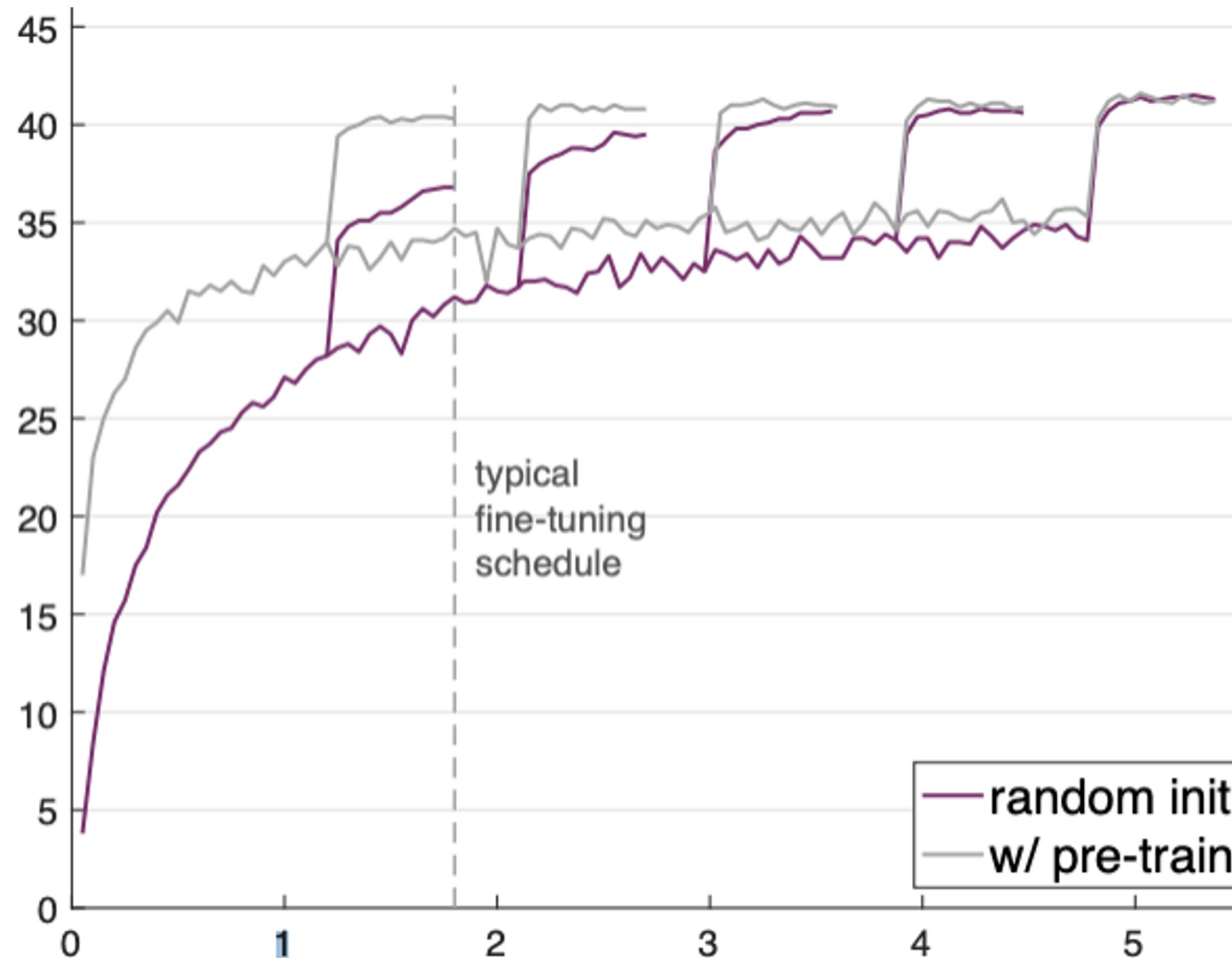
Improvements in CNN architecture leads to improvements in many downstream tasks thanks to transfer learning!

Transfer Learning: Architecture Matters!



Transfer Learning can help you converge faster

coco object detection



If you have enough data and train for much longer, random initialization can sometimes do as well as transfer learning

Transfer Learning is pervasive! It's the norm, not the exception



Very active area of research!

Call for papers

Important dates (all times AoE)

- Submissions open: Feb 15th 2023
- Submission deadline: Apr 14th 2023
- Decision notification: Apr 30th 2023
- Camera ready deadline: May 14th 2023
- Workshop: May 29th 2023

Classification: Transferring to New Tasks

Classification



"Chocolate Pretzels"

No spatial extent



Semantic Segmentation



Chocolate Pretzels,
Shelf

No objects, just pixels

Object Detection



Flipz, Hershey's, Keese's

Multiple objects

Instance Segmentation



Today: Object Detection

Classification



"Chocolate Pretzels"

No spatial extent



Semantic Segmentation



Chocolate Pretzels,
Shelf

No objects, just pixels

Object Detection



Flipz, Hershey's, Keese's

Multiple objects

Instance Segmentation



Object Detection: Task definition

Input: Single RGB image

Output: A set of detected objects;
For each object predict:

1. Category label (from a fixed set of labels)
2. Bounding box (four numbers: x, y, width, height)



Object Detection: Challenges

Multiple outputs: Need to output variable numbers of objects per image

Multiple types of output: Need to predict "what" (category label) as well as "where" (bounding box)

Large images: Classification works at 224x224; need higher resolution for detection, often ~800x600

1.



Bounding Boxes

Bounding boxes are typically axis-aligned

1.



Bounding Boxes

Bounding boxes are typically axis-aligned

1.

Oriented boxes are much less common

1.



Object Detection: Modal vs Amodal Boxes

Bounding boxes cover only the visible portion of the object

1.



Object Detection: Modal vs Amodal Boxes

Bounding boxes cover only the visible portion of the object

1.

Amodal detection: box covers the entire extent of the object, even occluded parts

1.



Object Detection: Modal vs Amodal Boxes

“Modal” detection: Bounding boxes (usually) cover only the visible portion of the object

1.

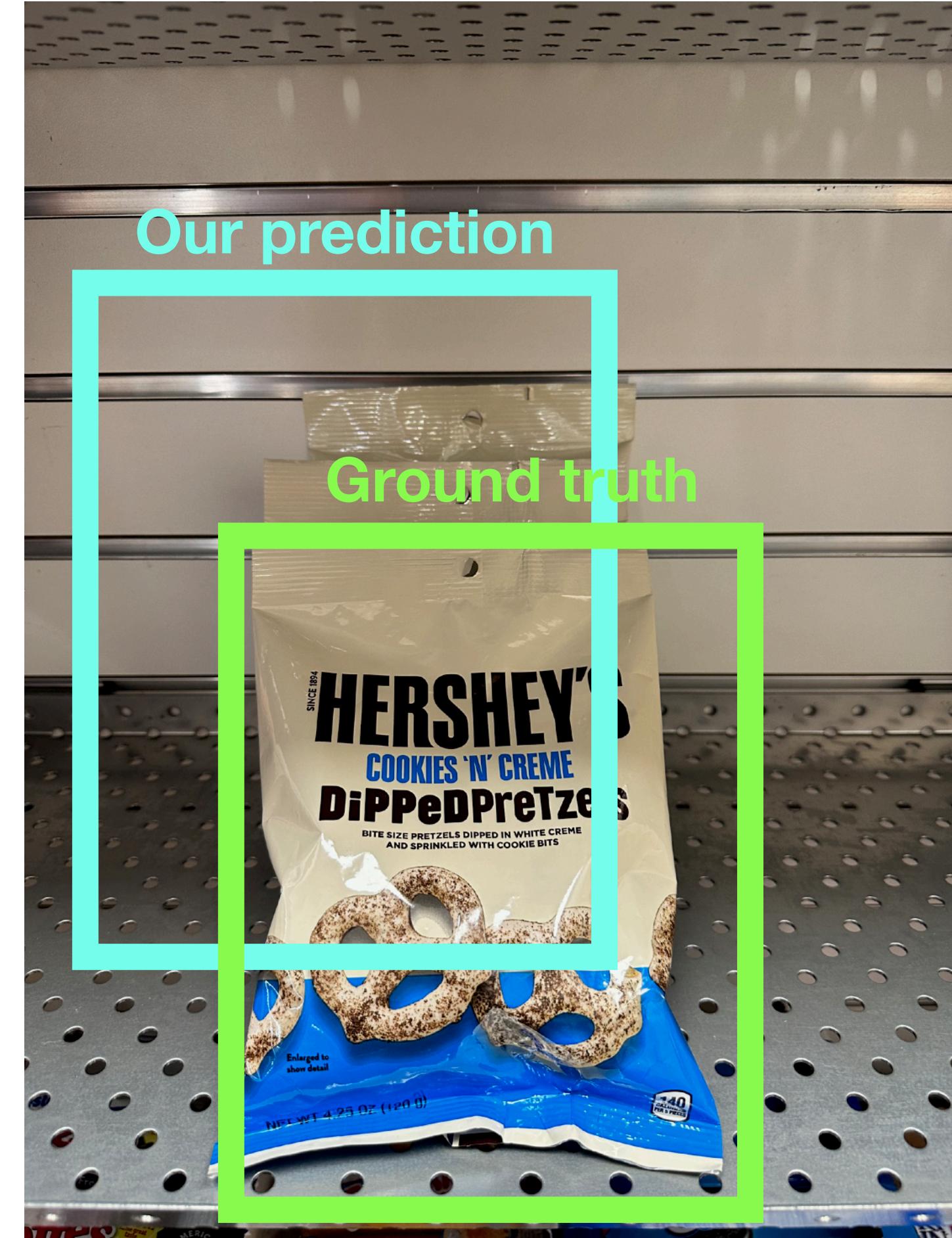
Amodal detection: box covers the entire extent of the object, even occluded parts

1.



Comparing Boxes: Intersection over Union (IoU)

How can we compare our prediction to the ground-truth box?



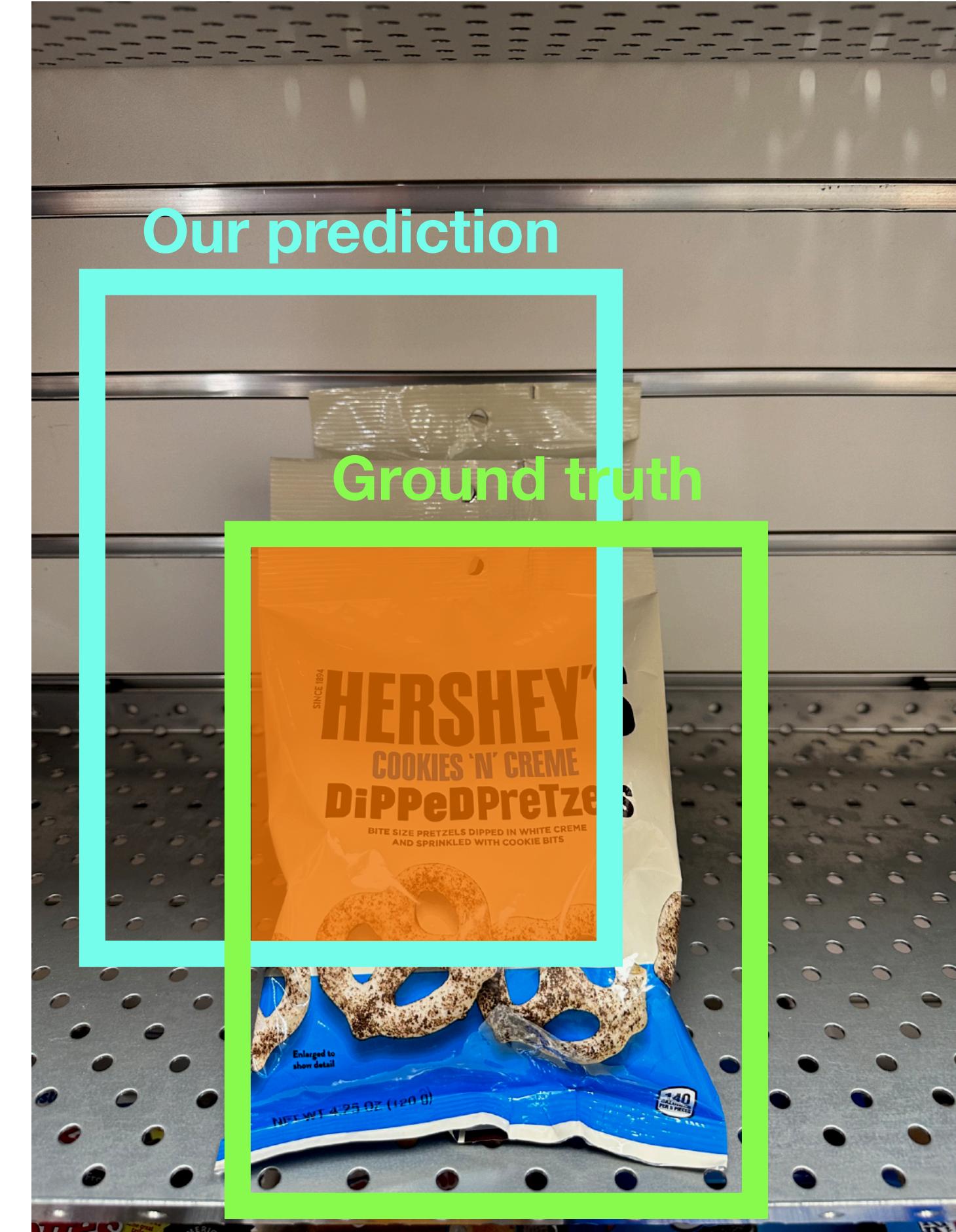
Comparing Boxes: Intersection over Union (IoU)

How can we compare our prediction to the ground-truth box?

Intersection over Union (IoU) (Also called “Jaccard similarity” or “Jaccard index”):

Area of Intersection

Area of Union



Comparing Boxes: Intersection over Union (IoU)

How can we compare our prediction to the ground-truth box?

Intersection over Union (IoU) (Also called “Jaccard similarity” or “Jaccard index”):

Area of Intersection

Area of Union



Comparing Boxes: Intersection over Union (IoU)

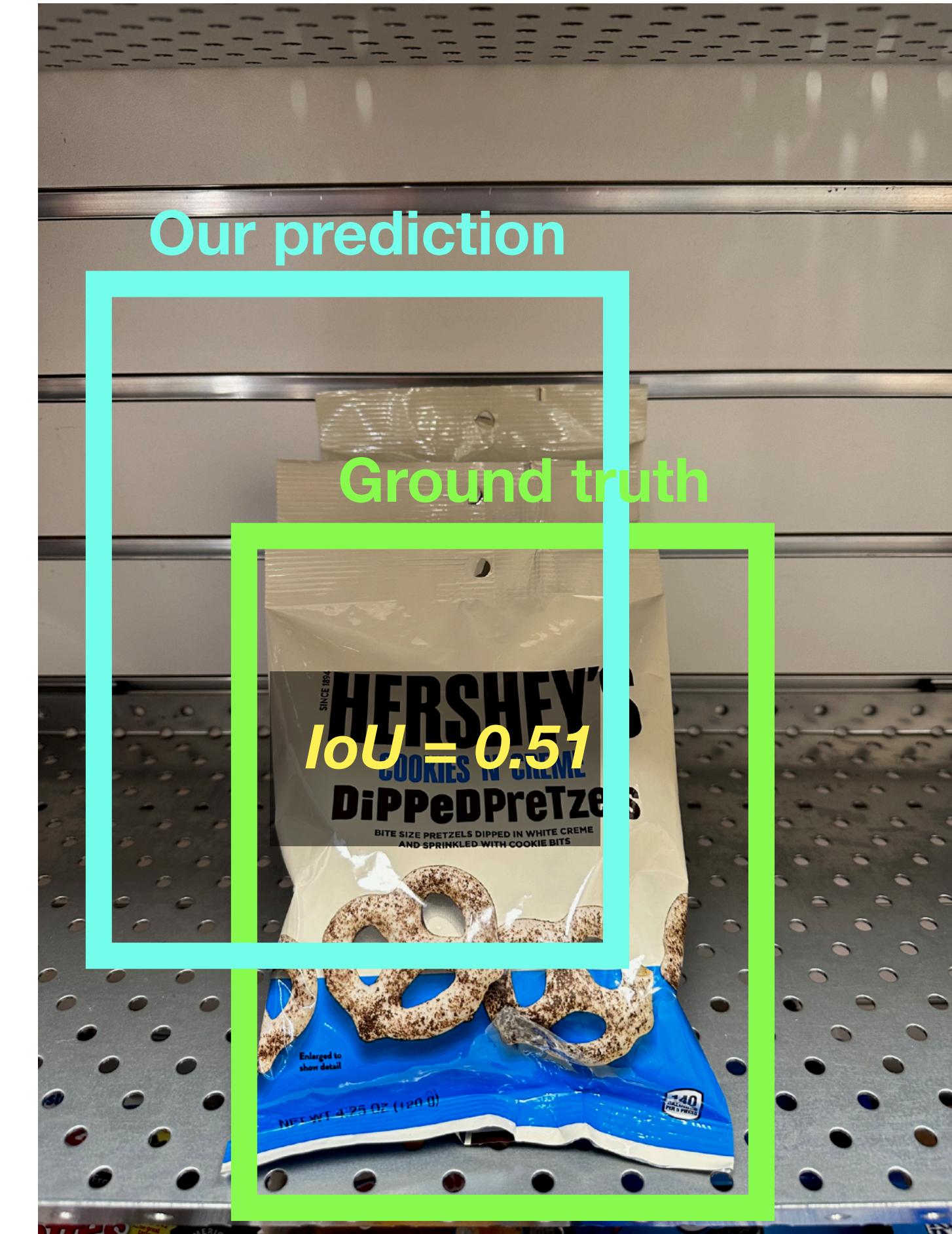
How can we compare our prediction to the ground-truth box?

Intersection over Union (IoU) (Also called “Jaccard similarity” or “Jaccard index”):

Area of Intersection

Area of Union

$\text{IoU} > 0.5$ is “decent”,



Comparing Boxes: Intersection over Union (IoU)

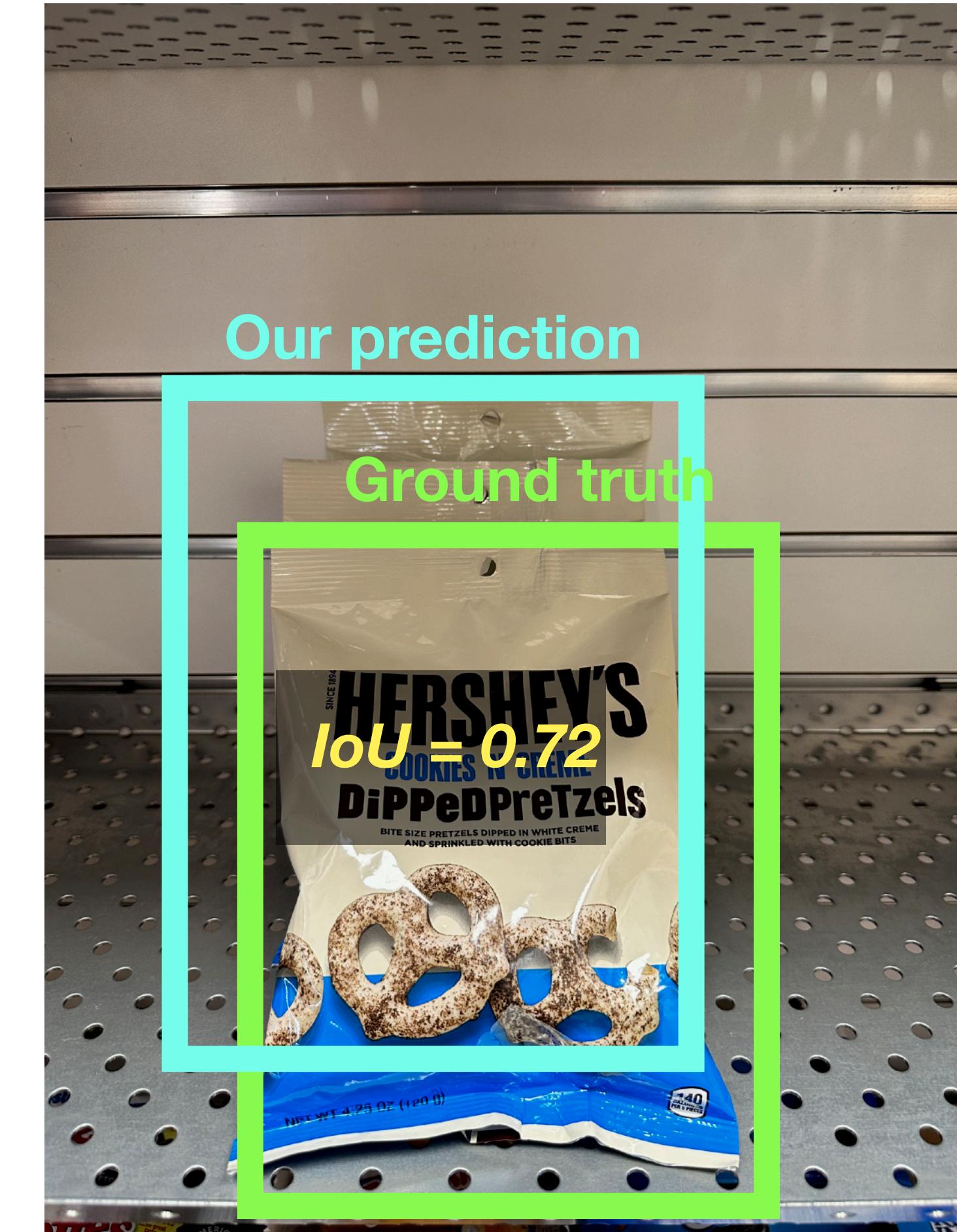
How can we compare our prediction to the ground-truth box?

Intersection over Union (IoU) (Also called “Jaccard similarity” or “Jaccard index”):

Area of Intersection

Area of Union

$\text{IoU} > 0.5$ is “decent”,
 $\text{IoU} > 0.7$ is “pretty good”,



Comparing Boxes: Intersection over Union (IoU)

How can we compare our prediction to the ground-truth box?

Intersection over Union (IoU) (Also called “Jaccard similarity” or “Jaccard index”):

Area of Intersection

Area of Union

$\text{IoU} > 0.5$ is “decent”,
 $\text{IoU} > 0.7$ is “pretty good”,
 $\text{IoU} > 0.9$ is “almost perfect”



Detecting a single object

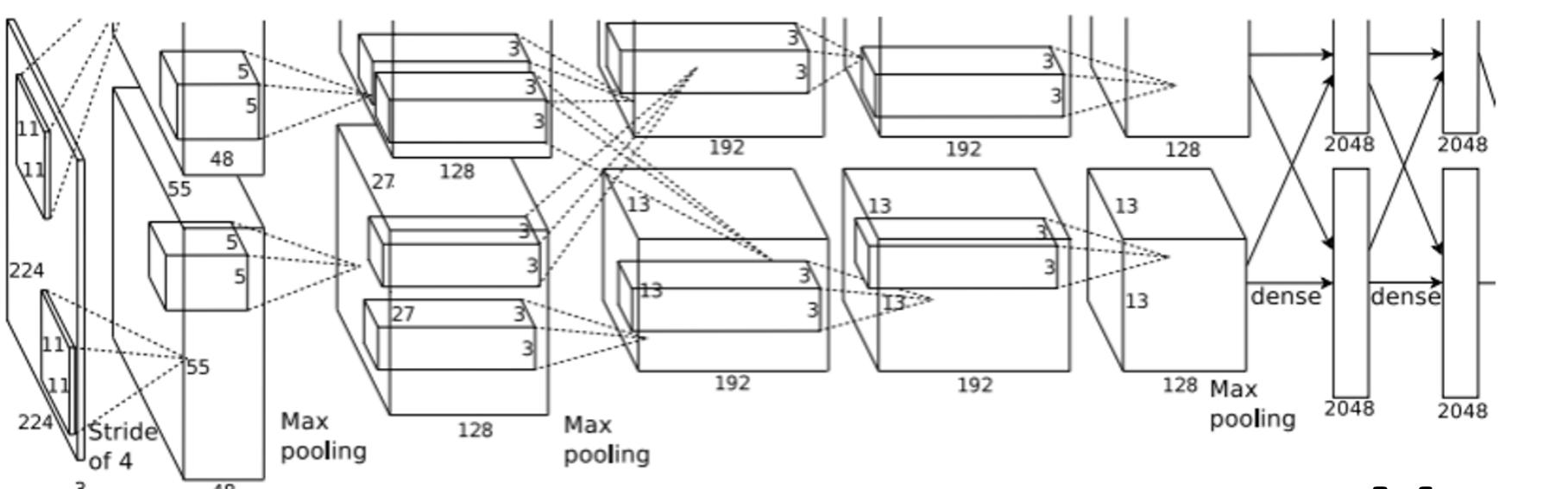


Figure copyright Alex Krizhevsky, Ilya Sutskever, and Geoffrey Hinton, 2012. Reproduced with permission.

Vector:
4096

Treat localization as a regression problem!

Loss

Detecting a single object

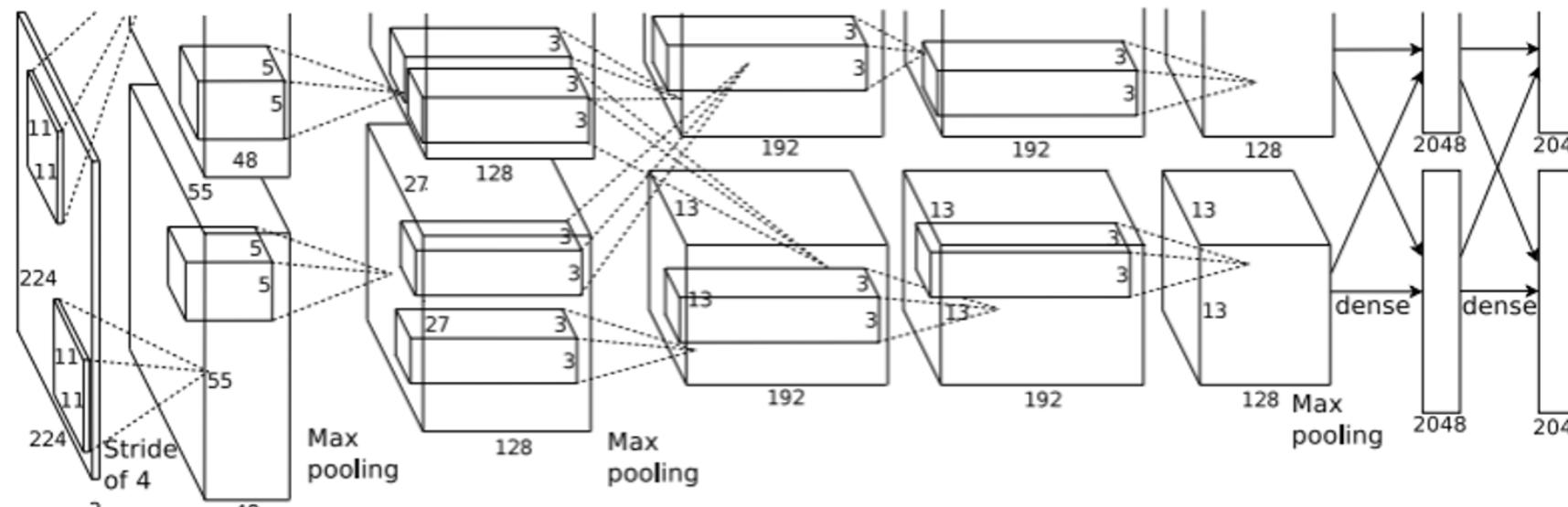


Figure copyright Alex Krizhevsky, Ilya Sutskever, and Geoffrey Hinton, 2012. Reproduced with permission.

Treat localization as a regression problem!

Fully connected:
4096 to 10

Vector:
4096

What??

Class scores:

Chocolate Pretzels: 0.9
Granola Bar: 0.02
Potato Chips: 0.02
Water Bottle: 0.02
Popcorn: 0.01
....

Correct Label:
Chocolate Pretzels

→ Softmax Loss

Detecting a single object

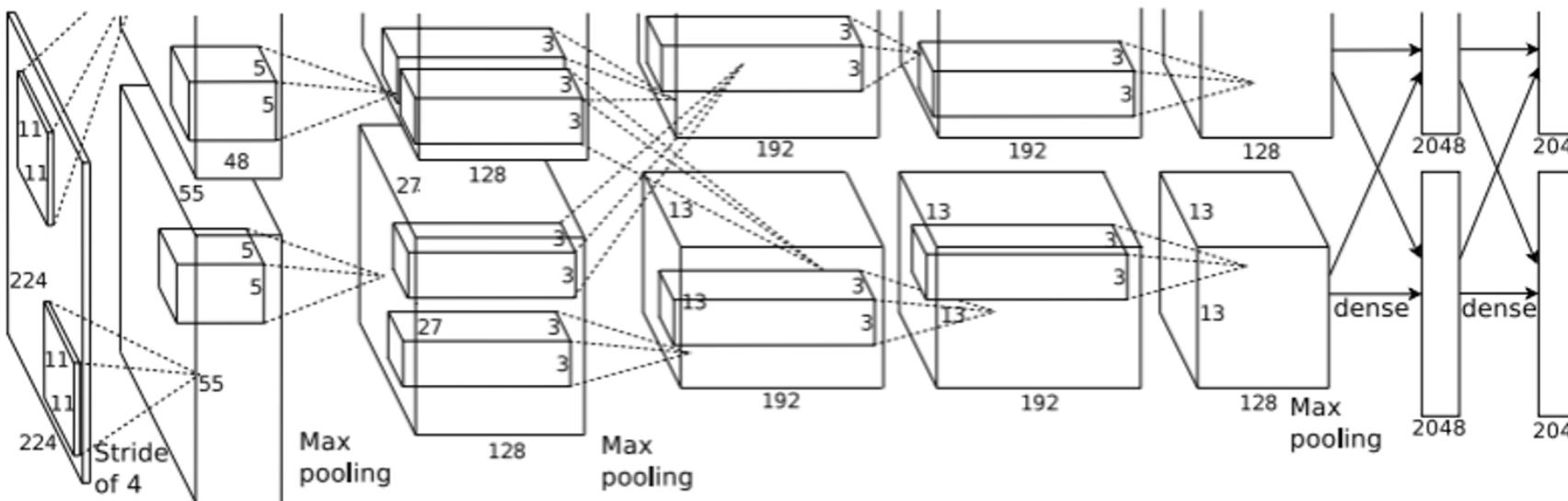
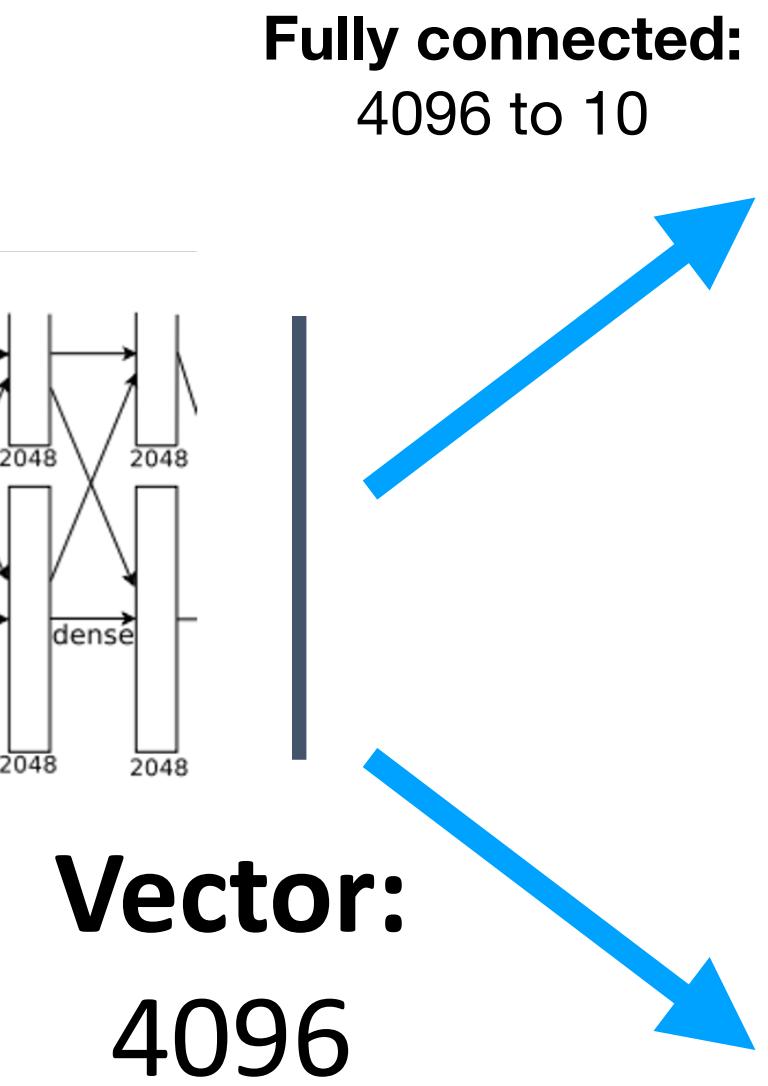


Figure copyright Alex Krizhevsky, Ilya Sutskever, and Geoffrey Hinton, 2012. Reproduced with permission.

Treat localization as a regression problem!



What??

Class scores:

Chocolate Pretzels: 0.9
Granola Bar: 0.02
Potato Chips: 0.02
Water Bottle: 0.02
Popcorn: 0.01
....

Correct Label:
Chocolate Pretzels

Softmax Loss

Where??

Box coordinates:
 (x, y, w, h)

L2 Loss

Correct coordinates:
 (x', y', w', h')

Detecting a single object

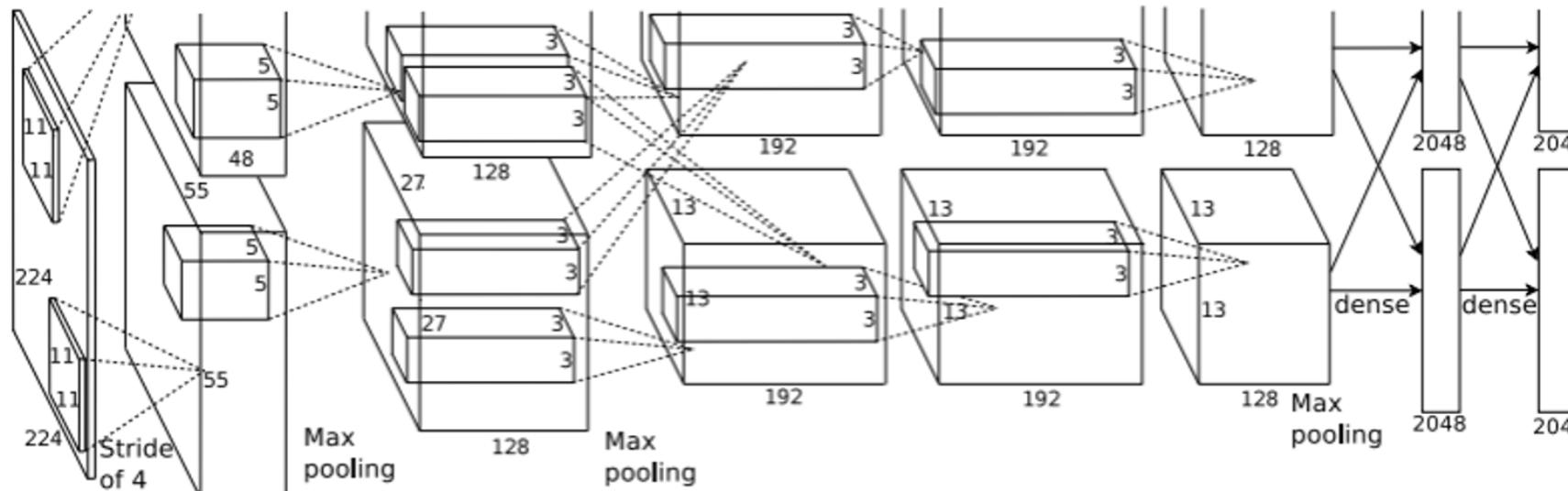


Figure copyright Alex Krizhevsky, Ilya Sutskever, and Geoffrey Hinton, 2012. Reproduced with permission.

Treat localization as a regression problem!



Fully connected:
4096 to 10

Vector:
4096

Fully connected:
4096 to 10

What??

Class scores:

Chocolate Pretzels: 0.9
Granola Bar: 0.02
Potato Chips: 0.02
Water Bottle: 0.02
Popcorn: 0.01
....

Correct Label:
Chocolate Pretzels

Softmax Loss

Multitask Loss

Loss

$$L = L_{cls} + \lambda L_{reg}$$

Box coordinates:
(x, y, w, h)

L2 Loss

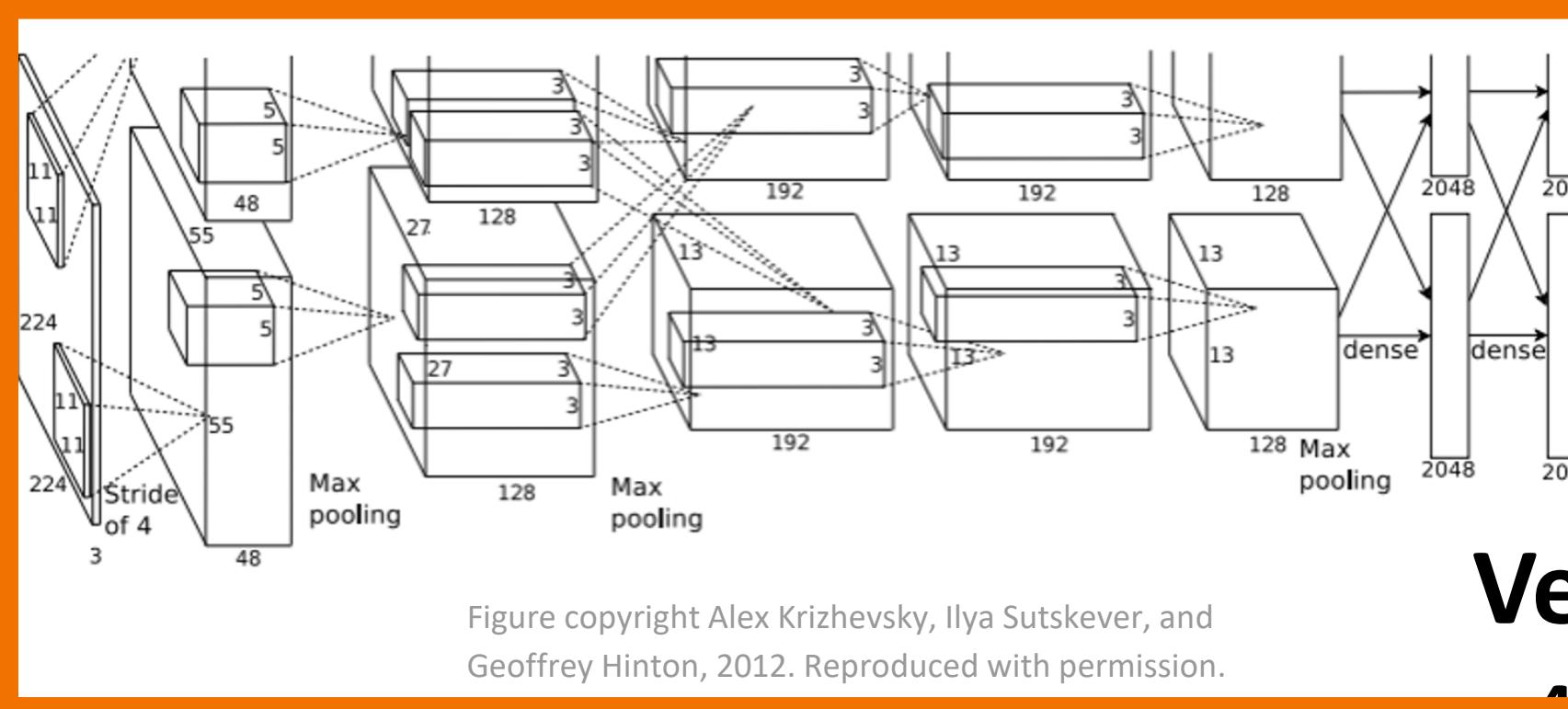
Where??

Correct coordinates:
(x', y', w', h')

Detecting a single object



Often pretrained on ImageNet: Transfer learning



Treat localization as a regression problem!



Fully connected: 4096 to 10
Vector: 4096

What??

Class scores:
Chocolate Pretzels: 0.9
Granola Bar: 0.02
Potato Chips: 0.02
Water Bottle: 0.02
Popcorn: 0.01
....

Correct Label:
Chocolate Pretzels

Softmax Loss

Multitask Loss

Weighted Sum

Loss

$$L = L_{cls} + \lambda L_{reg}$$

Box coordinates:
(x, y, w, h)

L2 Loss

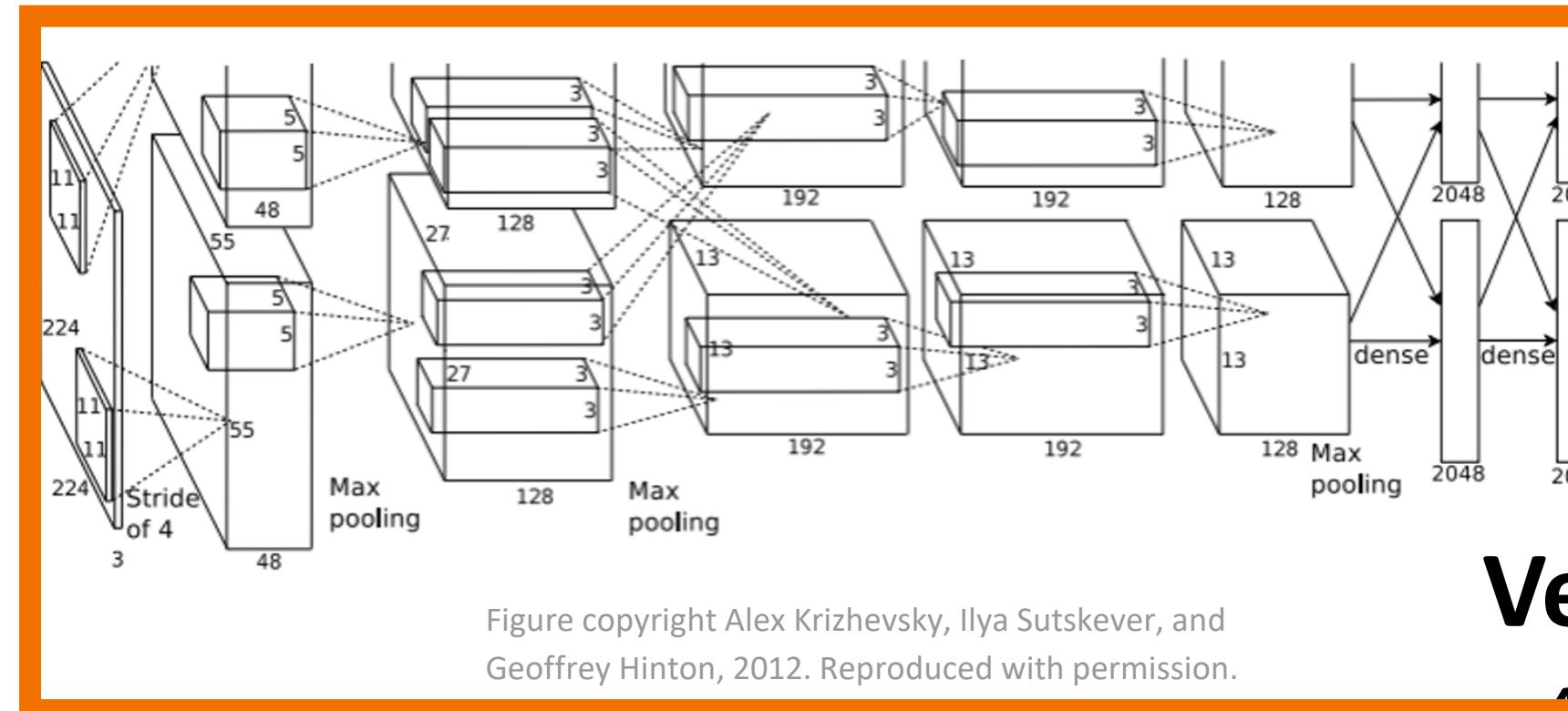
Where??

Correct coordinates:
(x', y', w', h')

Detecting a single object



Often pretrained on ImageNet: Transfer learning



Vector:
4096

Fully connected:
4096 to 10

Treat localization as a regression problem!

Problem: Images can have more than one object!

What??

Class scores:

Chocolate Pretzels: 0.9
Granola Bar: 0.02
Potato Chips: 0.02
Water Bottle: 0.02
Popcorn: 0.01
....

Correct Label:

Chocolate Pretzels

Softmax Loss

Multitask Loss

Loss

$$L = L_{cls} + \lambda L_{reg}$$

Weighted Sum

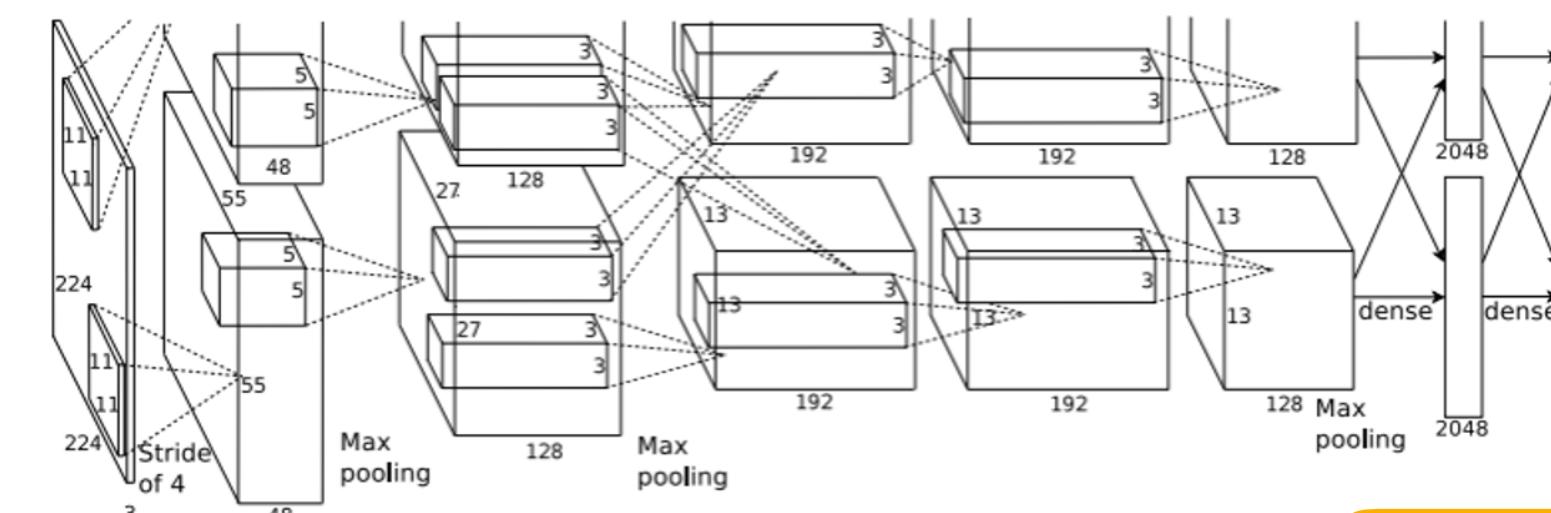
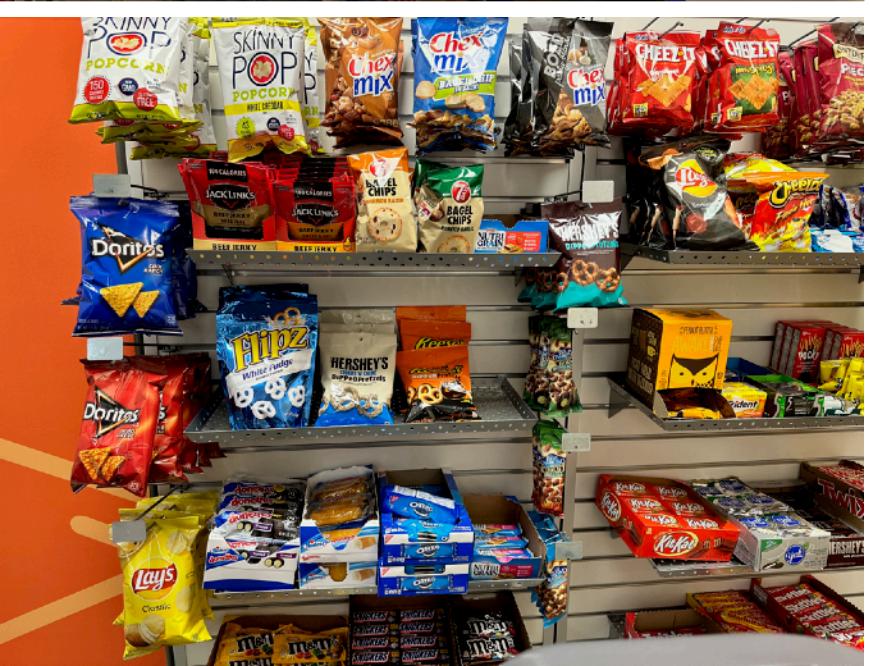
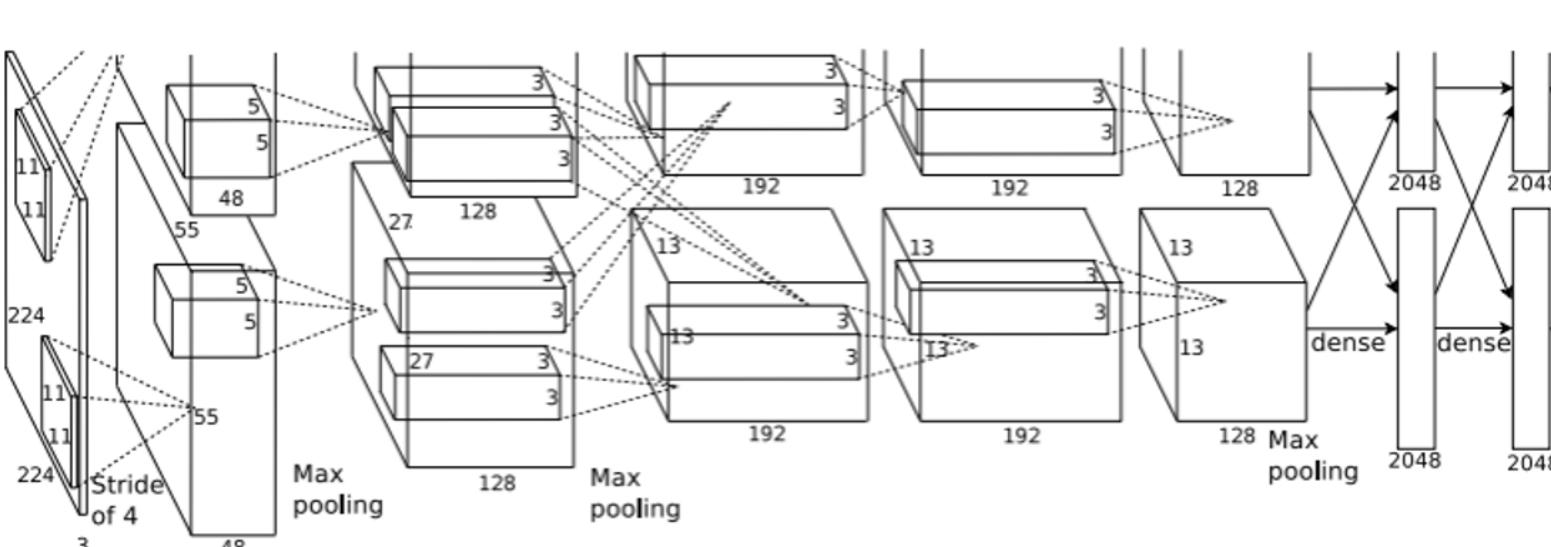
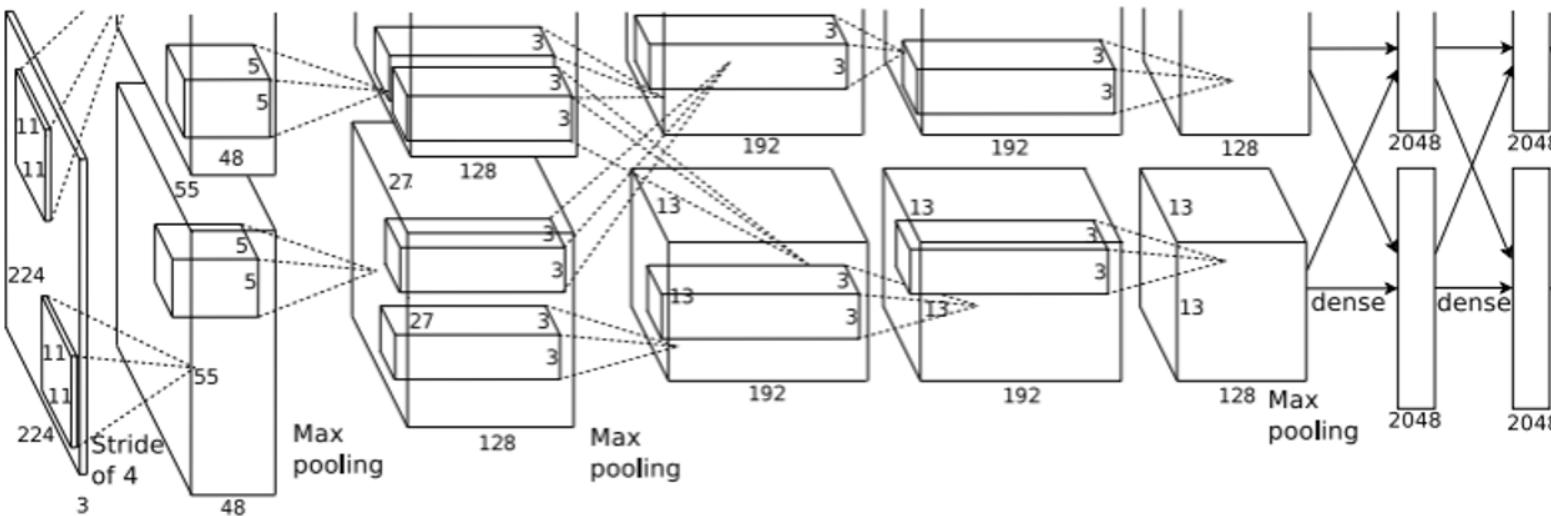
Box coordinates:
(x, y, w, h)

L2 Loss

Where??

Correct coordinates:
(x', y', w', h')

Detecting Multiple Objects



Hershey's: (x, y, w, h)
4 numbers

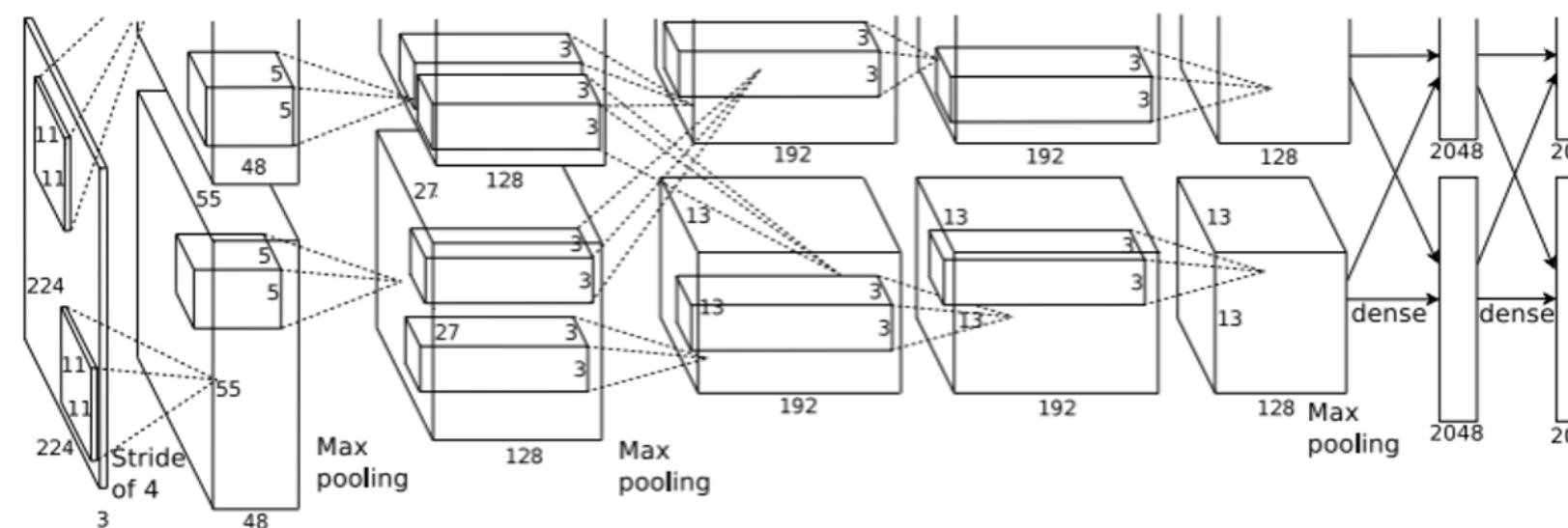
Hershey's: (x, y, w, h)
Flipz: (x, y, w, h)
Reese's (x, y, w, h)
12 numbers

Chips: (x, y, w, h)
Chips: (x, y, w, h)
.....
Many numbers!

**Need different numbers of
output per image**

Detecting Multiple Objects: Sliding Window

Apply a CNN to many different crops of the image, CNN classifies each crop as object or background

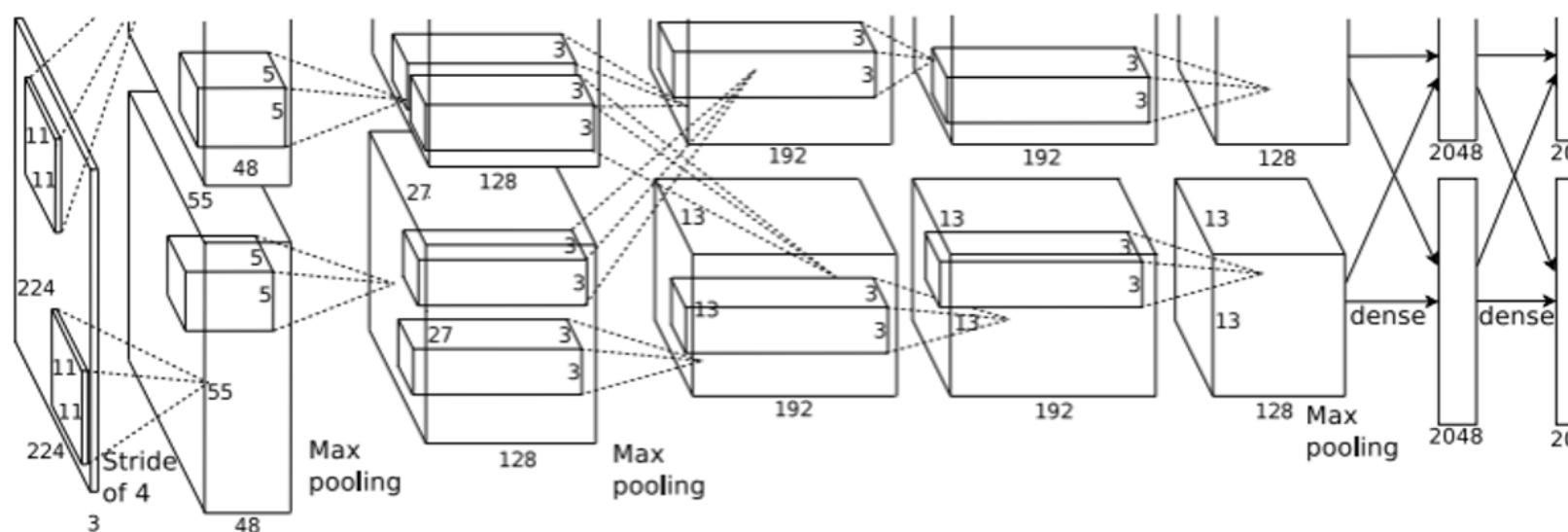


Hershey's: No
Flipz: No
Reese's: No
Background: Yes

Detecting Multiple Objects: Sliding Window



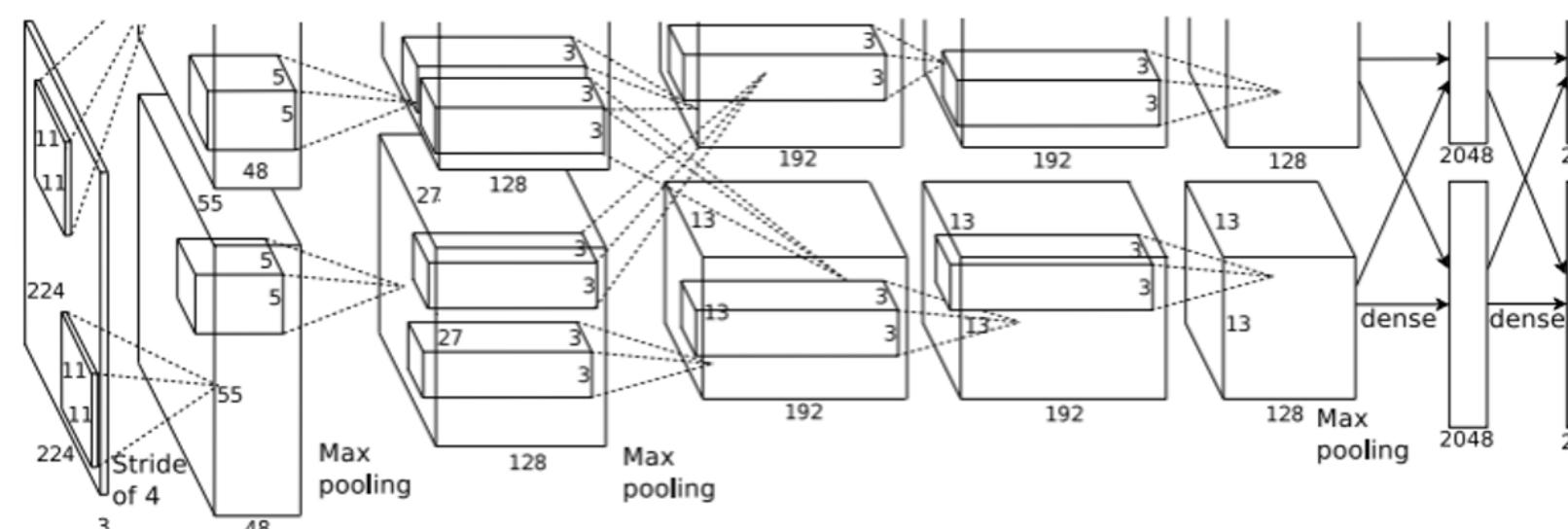
Apply a CNN to many different crops of the image, CNN classifies each crop as object or background



Hershey's: No
Flipz: Yes
Reese's: No
Background: No

Detecting Multiple Objects: Sliding Window

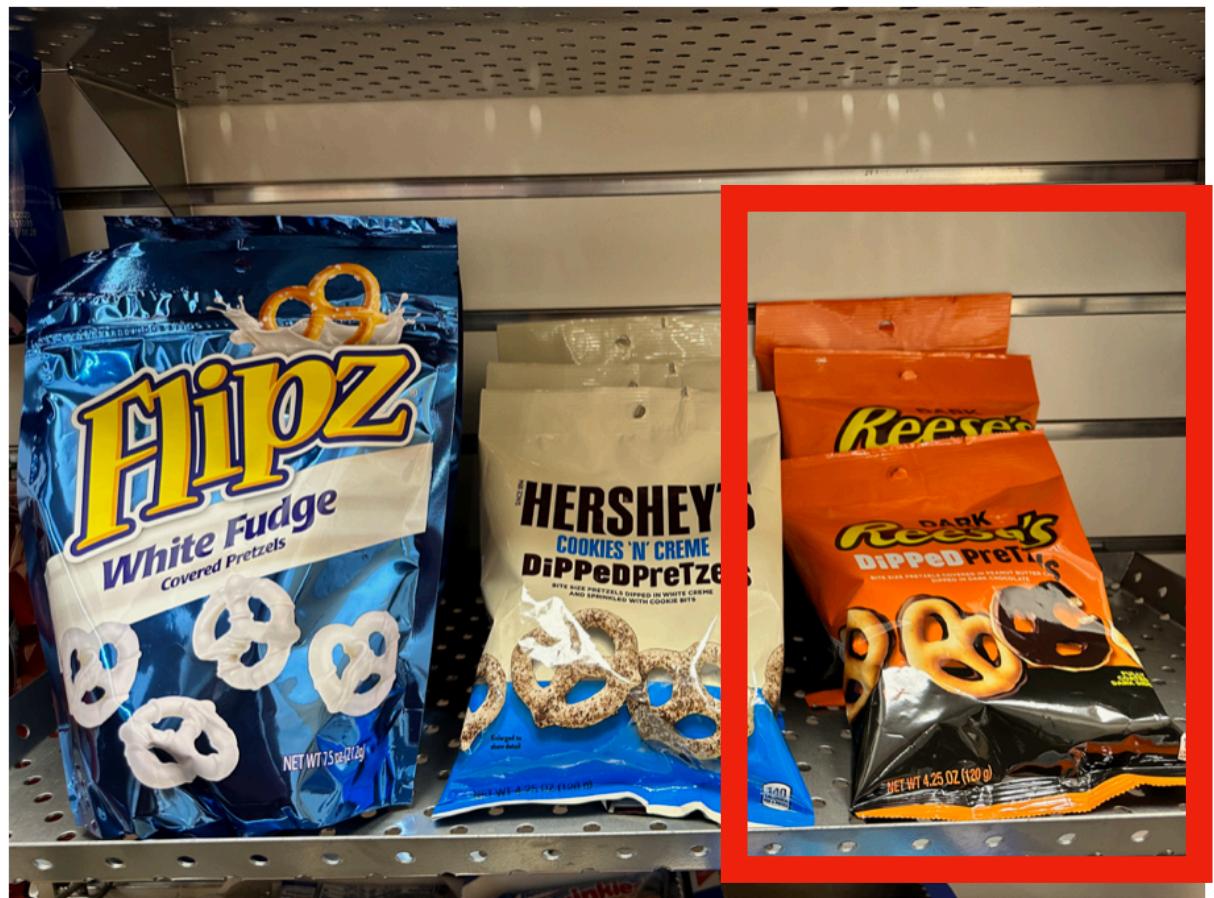
Apply a CNN to many different crops of the image, CNN classifies each crop as object or background



Hershey's: No
Flipz: No
Reese's: Yes
Background: No

Detecting Multiple Objects: Sliding Window

Apply a CNN to many different crops of the image, CNN classifies each crop as object or background



Question: How many possible boxes are there in an image of size $H \times W$?

Consider box of size $h \times w$:

Possible x positions: $W - w + 1$

Possible y positions: $H - h + 1$

Possible positions:

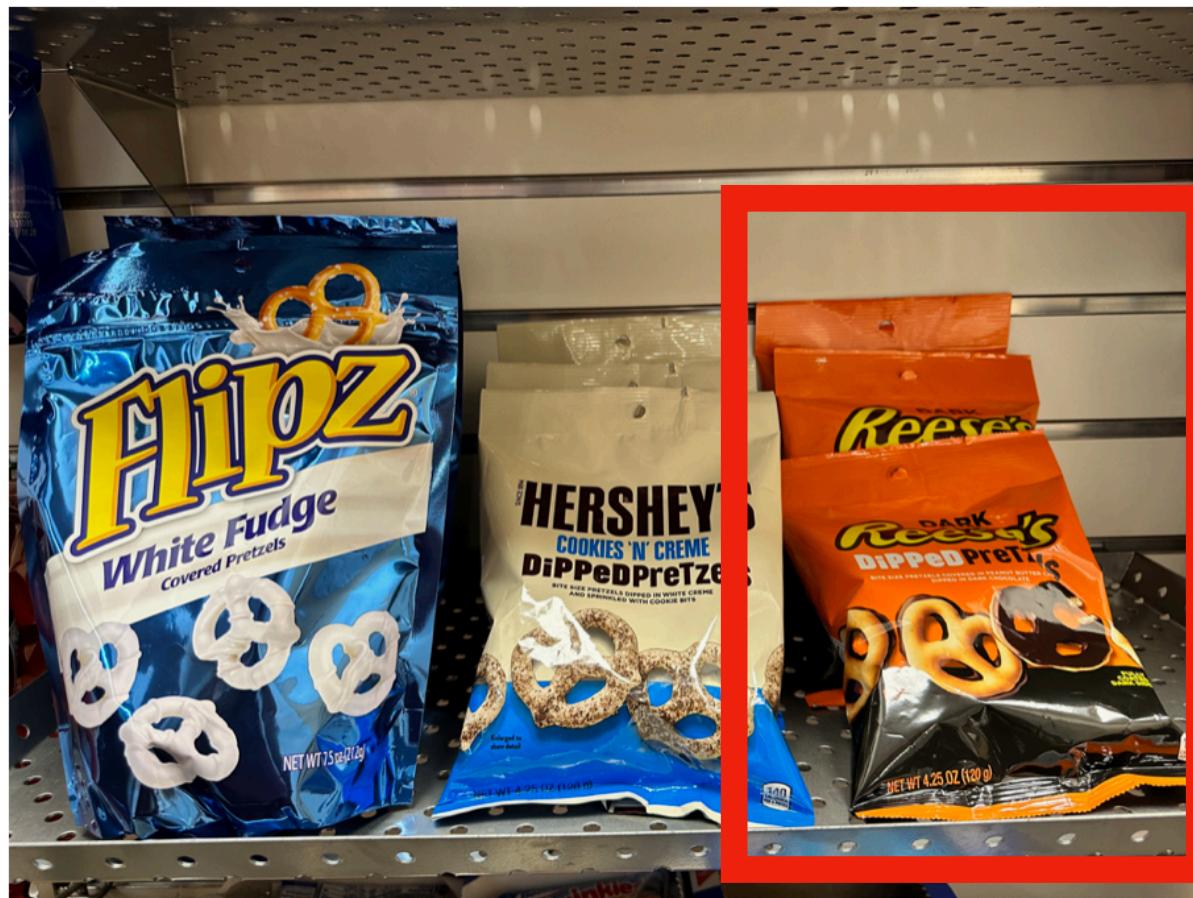
$(W-w+1) \times (H-h+1)$

Total possible boxes:

$$\sum_{h=1}^H \sum_{w=1}^W (W - w + 1)(H - h + 1)$$

$$= \frac{H(H+1)}{2} \frac{W(W+1)}{2}$$

Detecting Multiple Objects: Sliding Window



Apply a CNN to many different crops of the image, CNN classifies each crop as object or background

Question: How many possible boxes are there in an image of size $H \times W$?

Consider box of size $h \times w$:

Possible x positions: $W - w + 1$

Possible y positions: $H - h + 1$

Possible positions:

$(W-w+1) \times (H-h+1)$

800 x 600 image has
~58M boxes. No way
we can evaluate them
all

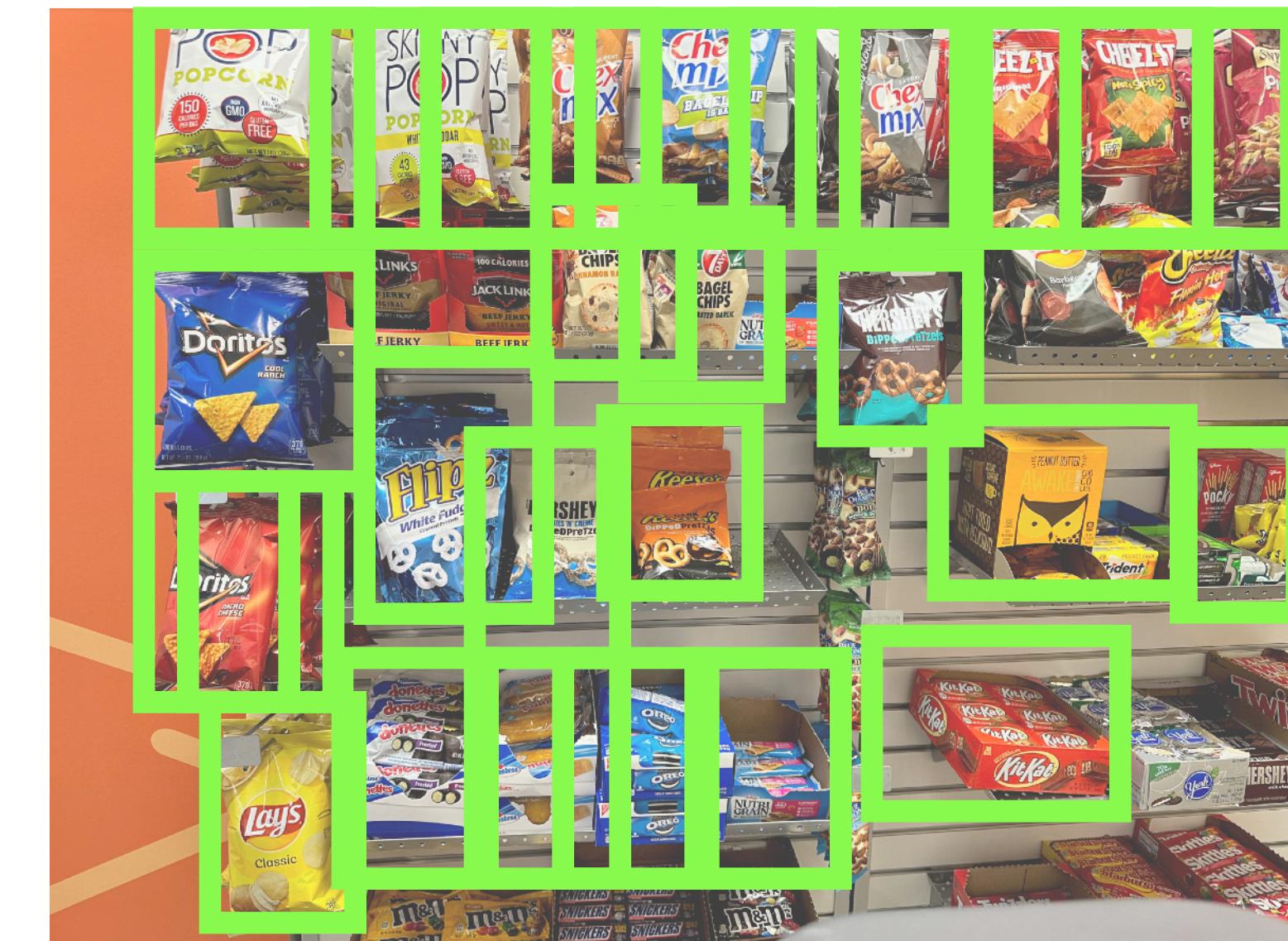
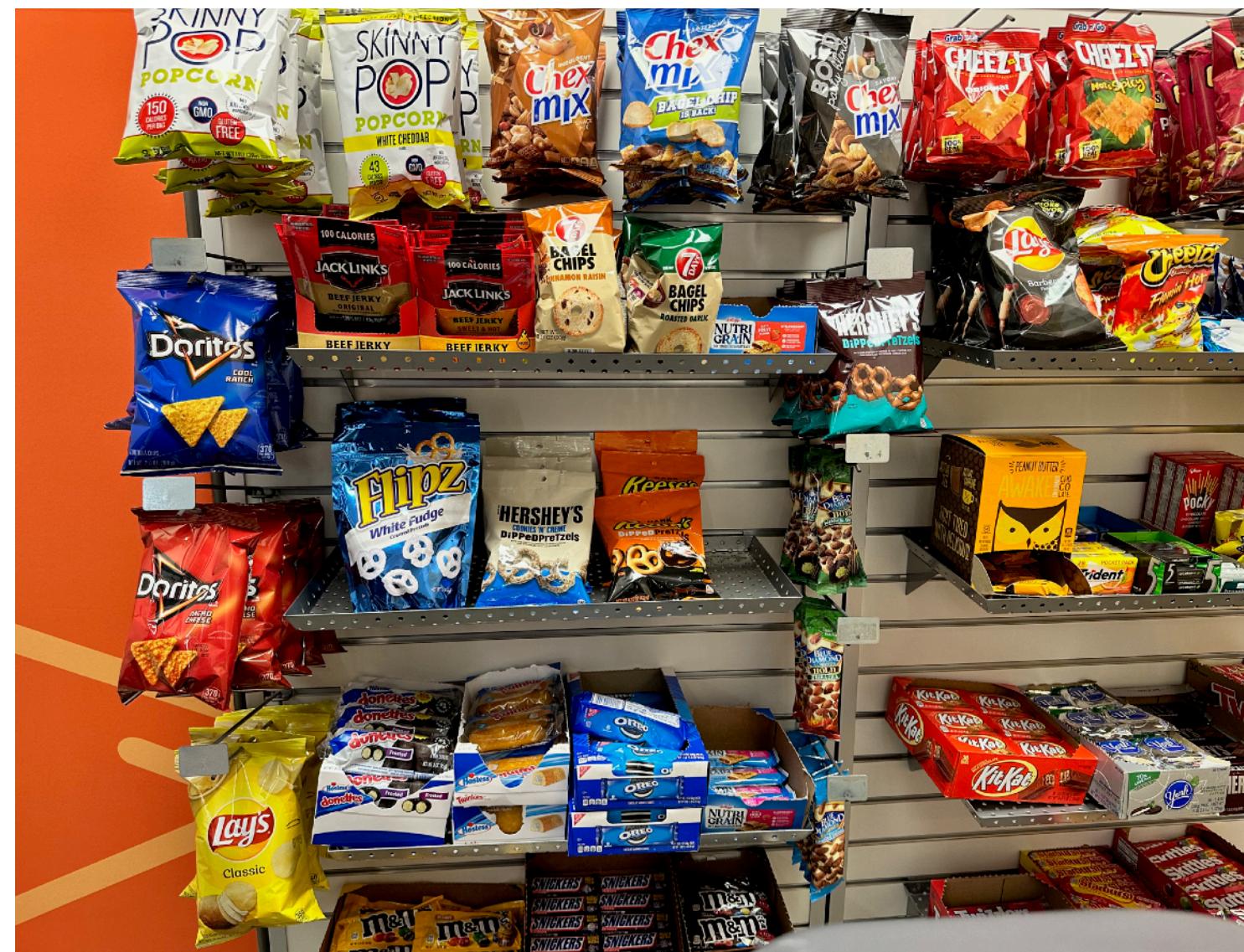
Total possible boxes:

$$\sum_{h=1}^H \sum_{w=1}^W (W - w + 1)(H - h + 1)$$

$$= \frac{H(H+1)}{2} \frac{W(W+1)}{2}$$

Region Proposals

- Find a small set of boxes that are likely to cover all objects
- Often based on heuristics: e.g. look for “blob-like” image regions
- Relatively fast to run; e.g. Selective Search gives 2000 region proposals in a few seconds on CPU



Alexe et al, “Measuring the objectness of image windows”, TPAMI 2012

Uijlings et al, “Selective Search for Object Recognition”, IJCV 2013

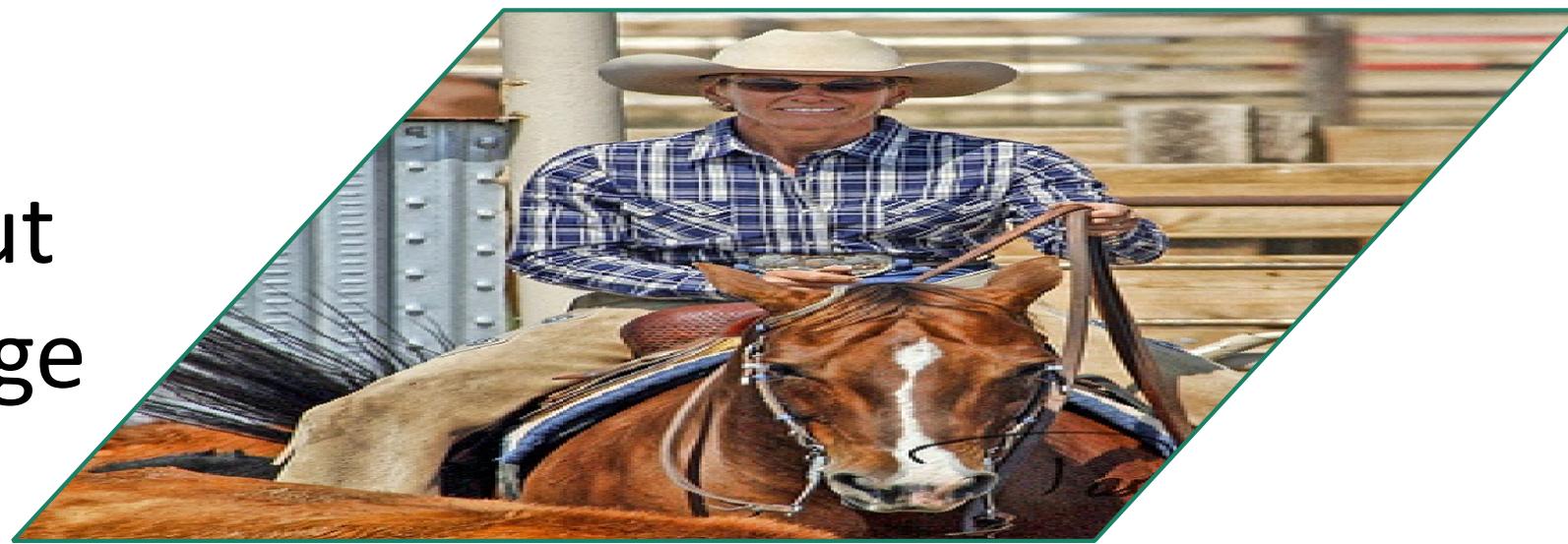
Cheng et al, “BING: Binarized normed gradients for objectness estimation at 300fps”, CVPR 2014

Zitnick and Dollar, “Edge boxes: Locating object proposals from edges”, ECCV 2014

R-CNN: Region-Based CNN

R-CNN: Region-Based CNN

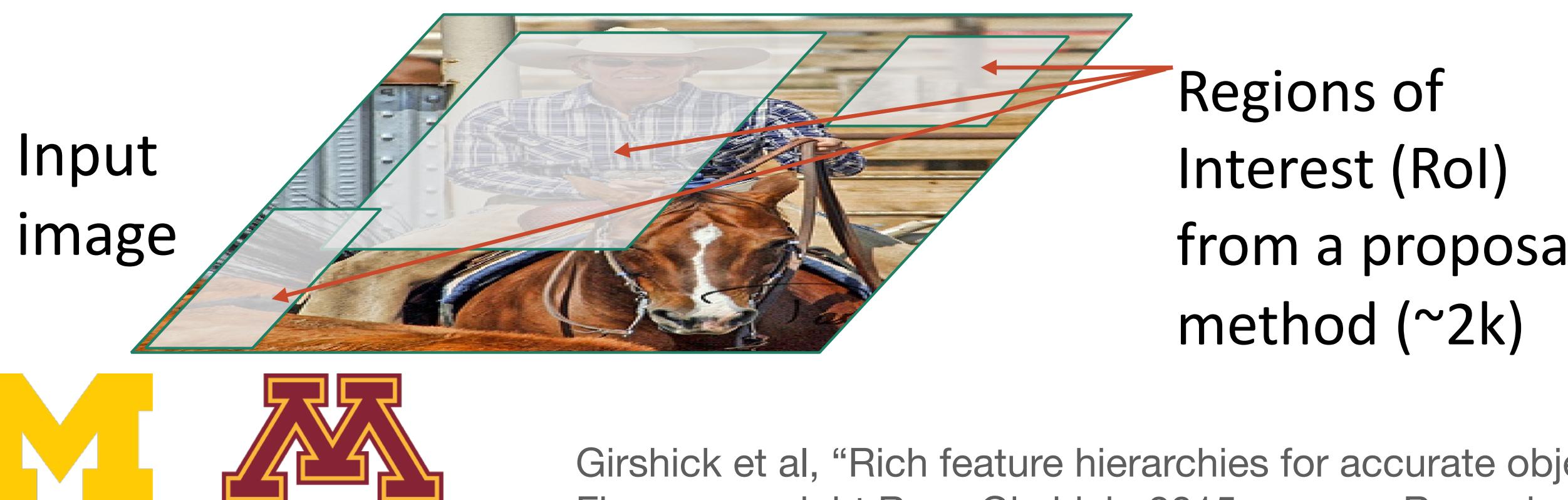
Input
image



Girshick et al, "Rich feature hierarchies for accurate object detection and semantic segmentation", CVPR 2014.
Figure copyright Ross Girshick, 2015; [source](#). Reproduced with permission

R-CNN: Region-Based CNN

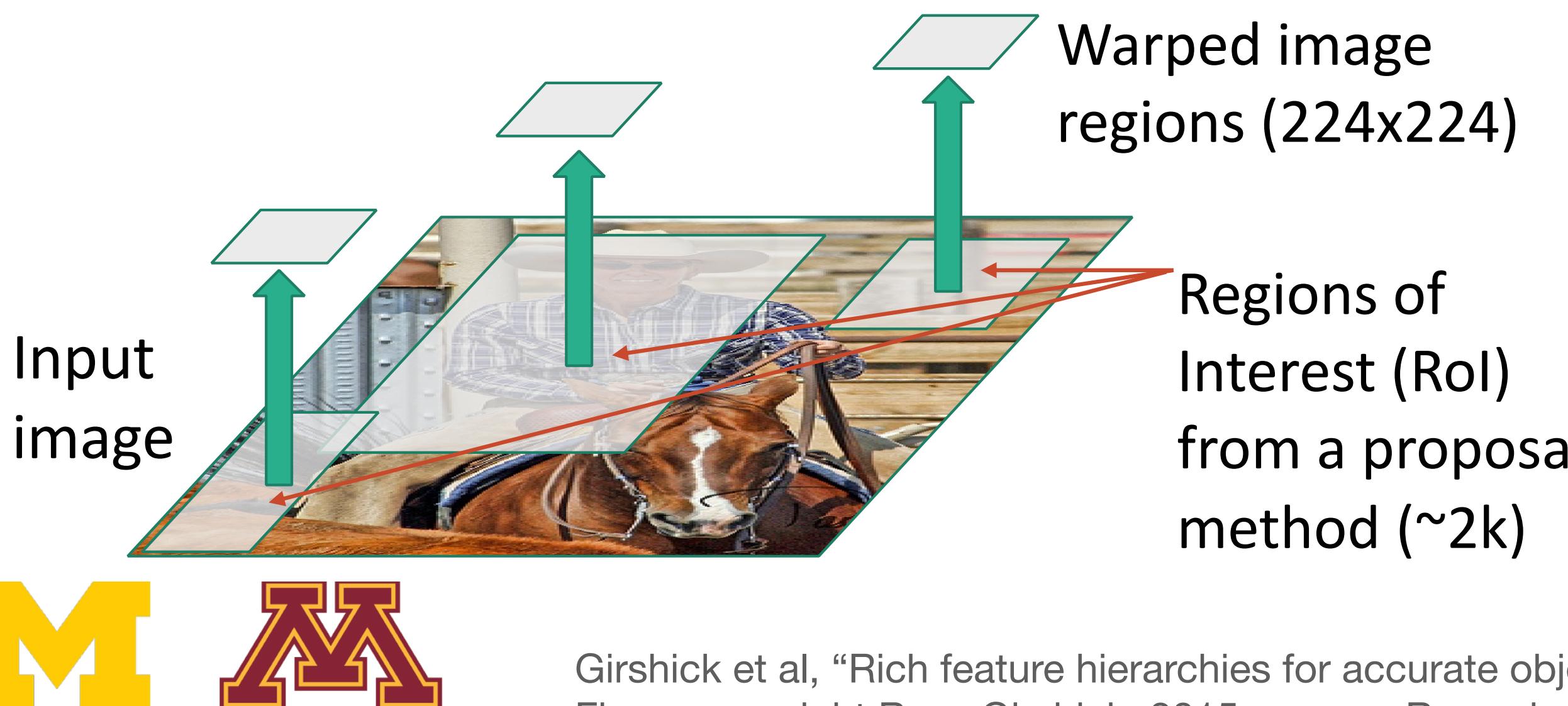
R-CNN: Region-Based CNN



Girshick et al, "Rich feature hierarchies for accurate object detection and semantic segmentation", CVPR 2014.
Figure copyright Ross Girshick, 2015; [source](#). Reproduced with permission

R-CNN: Region-Based CNN

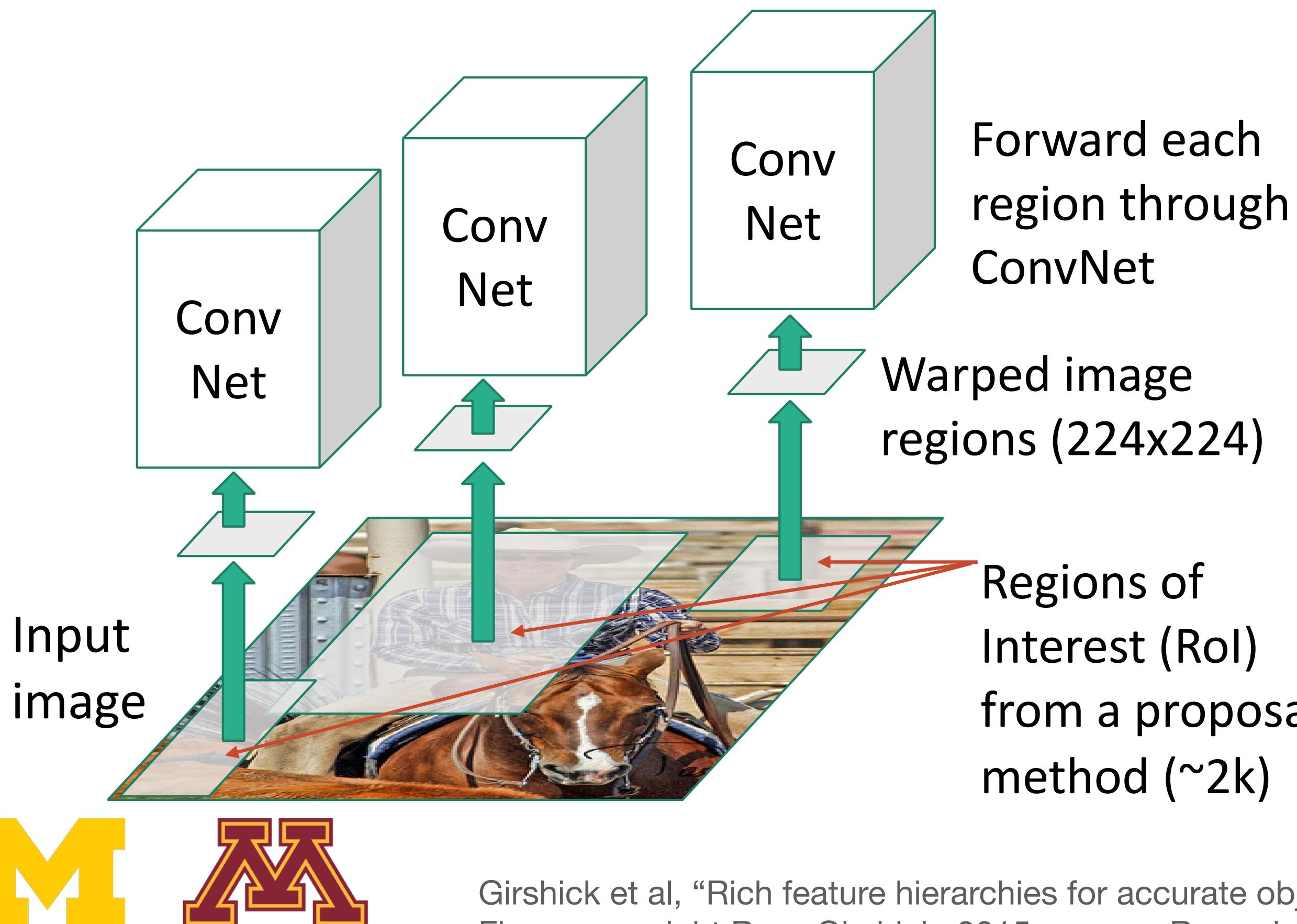
R-CNN: Region-Based CNN



Girshick et al, "Rich feature hierarchies for accurate object detection and semantic segmentation", CVPR 2014.
Figure copyright Ross Girshick, 2015; [source](#). Reproduced with permission

R-CNN: Region-Based CNN

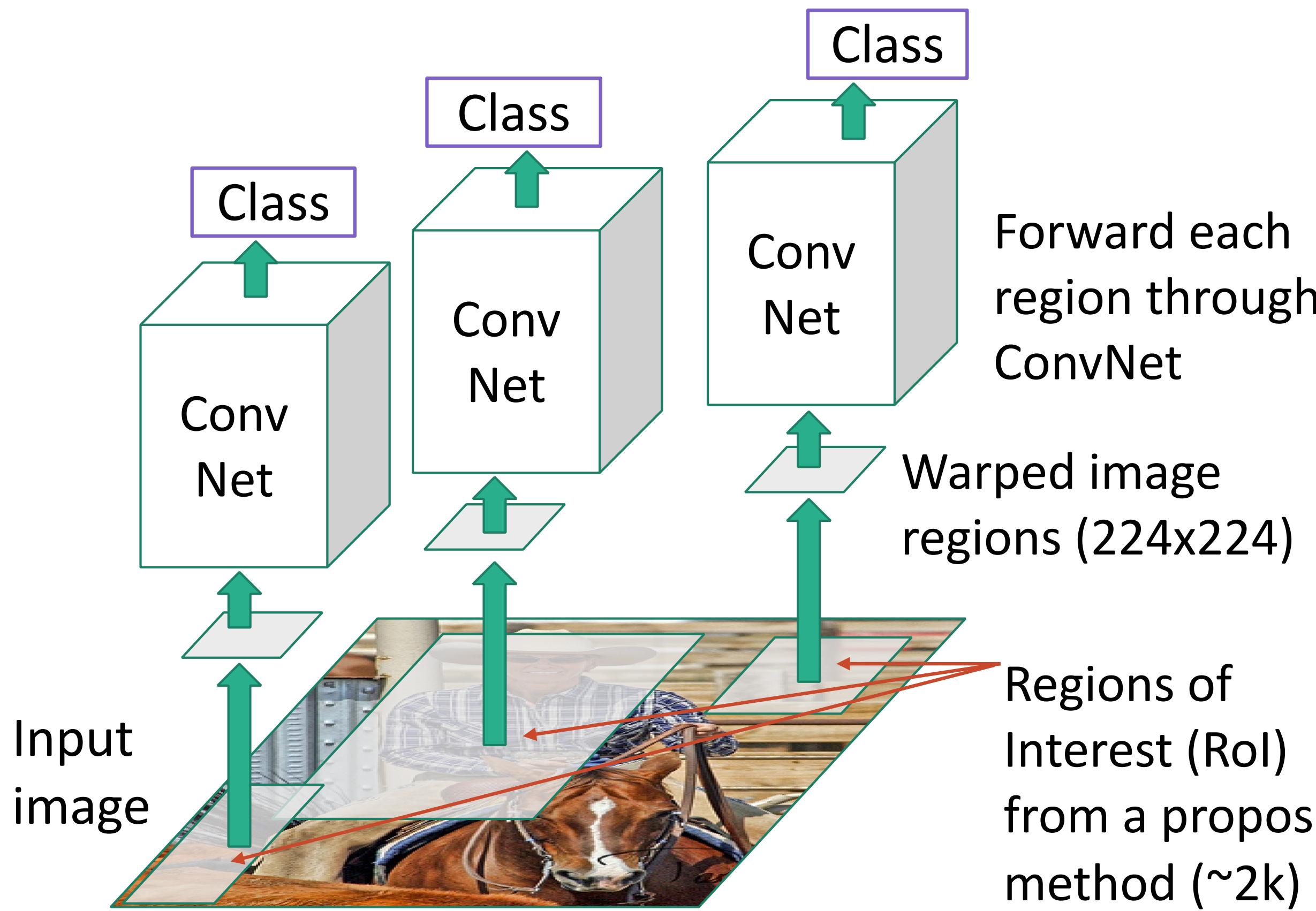
R-CNN: Region-Based CNN



Girshick et al, "Rich feature hierarchies for accurate object detection and semantic segmentation", CVPR 2014.
Figure copyright Ross Girshick, 2015; [source](#). Reproduced with permission

R-CNN: Region-Based CNN

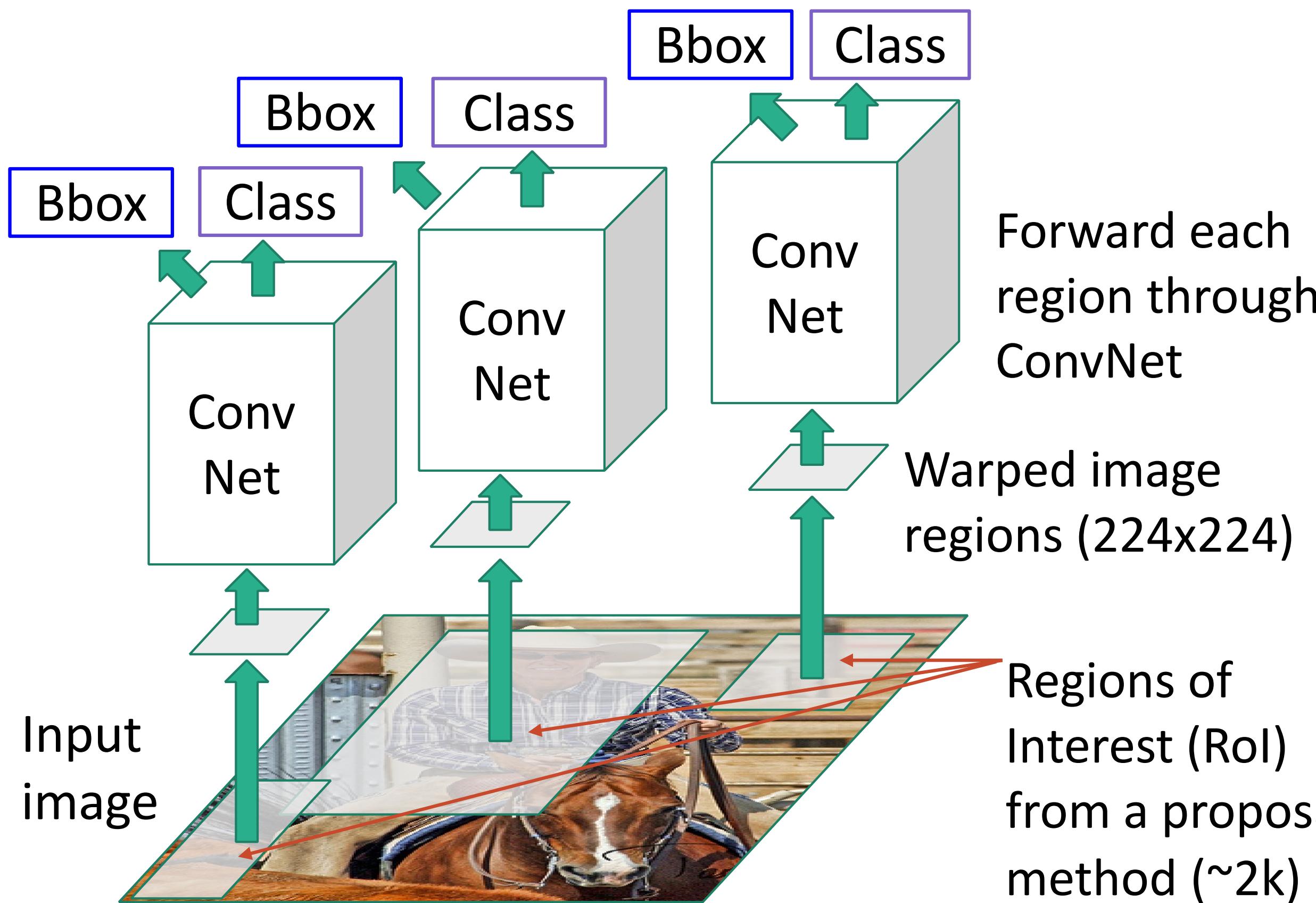
R-CNN: Region-Based CNN



Classify each region

R-CNN: Region-Based CNN

R-CNN: Region-Based CNN

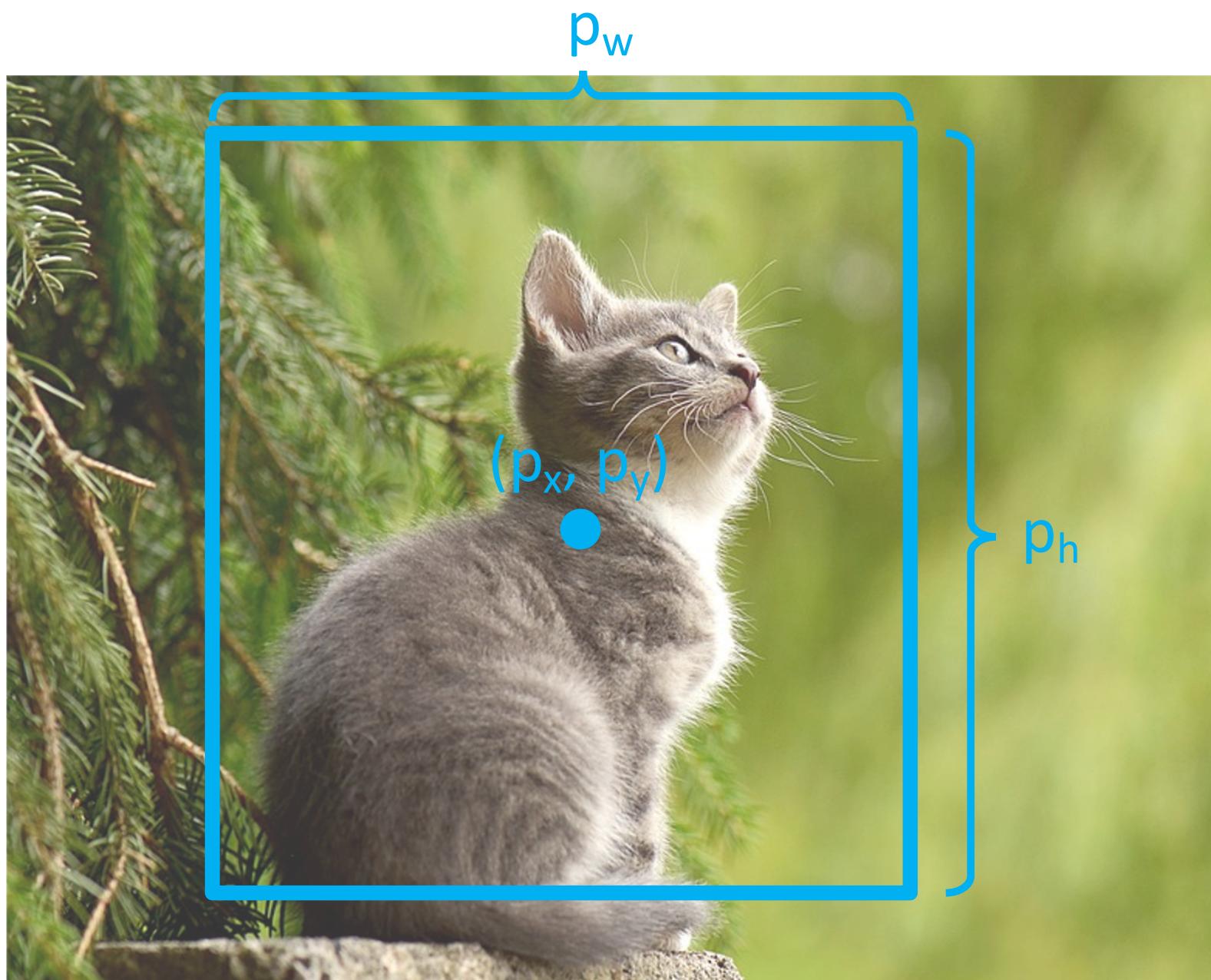


Classify each region

Bounding box regression:
Predict “transform” to correct the RoI: 4 numbers (t_x, t_y, t_h, t_w)

R-CNN: Box Regression

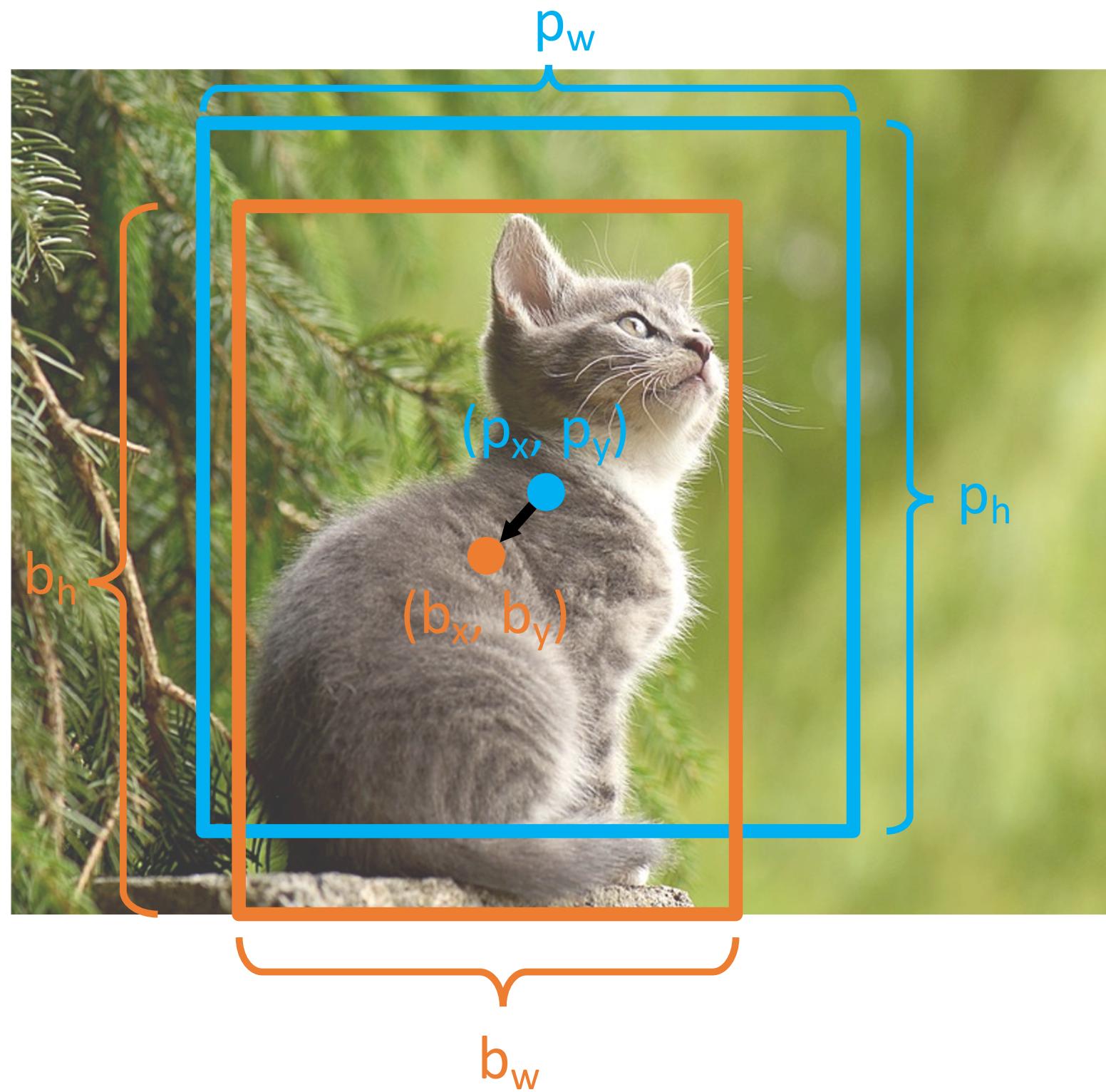
Consider a **region proposal** with center (p_x, p_y) , width p_w , height p_h



Model predicts a **transform** (t_x, t_y, t_w, t_h) to correct the region proposal

R-CNN: Box Regression

Consider a **region proposal** with center (p_x, p_y) , width p_w , height p_h



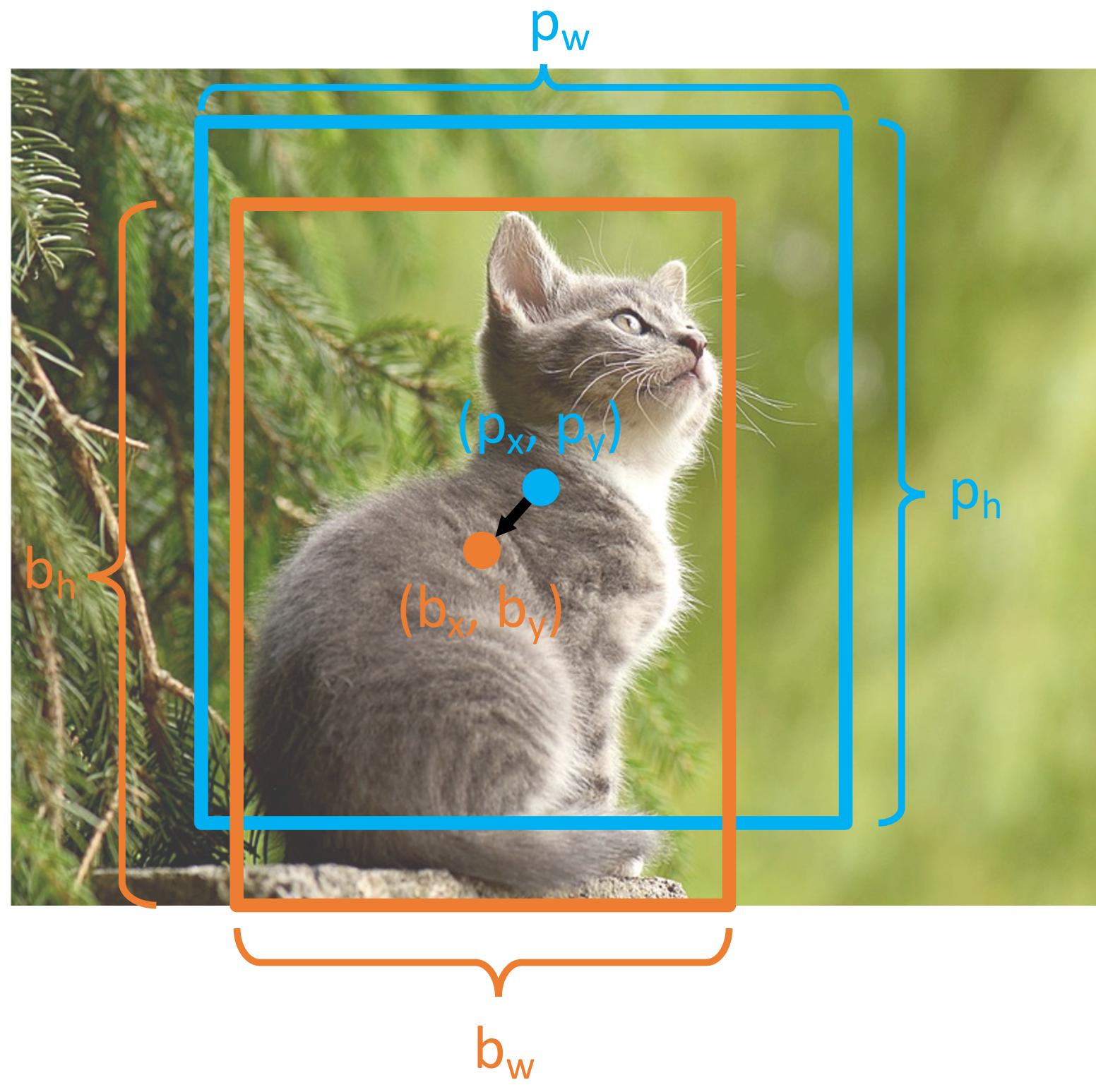
Model predicts a **transform** (t_x, t_y, t_w, t_h) to correct the region proposal

The **output box** is defined by:

$$\begin{aligned} b_x &= p_x + p_w t_x && \text{Shift center by amount relative to proposal size} \\ b_y &= p_y + p_h t_y \\ b_w &= p_w \exp(t_w) && \text{Scale proposal; exp ensures that scaling factor is } > 0 \\ b_h &= p_h \exp(t_h) \end{aligned}$$

R-CNN: Box Regression

Consider a **region proposal** with center (p_x, p_y) , width p_w , height p_h



Model predicts a **transform** (t_x, t_y, t_w, t_h) to correct the region proposal

The **output box** is defined by:

$$b_x = p_x + p_w t_x$$

$$b_y = p_y + p_h t_y$$

$$b_w = p_w \exp(t_w)$$

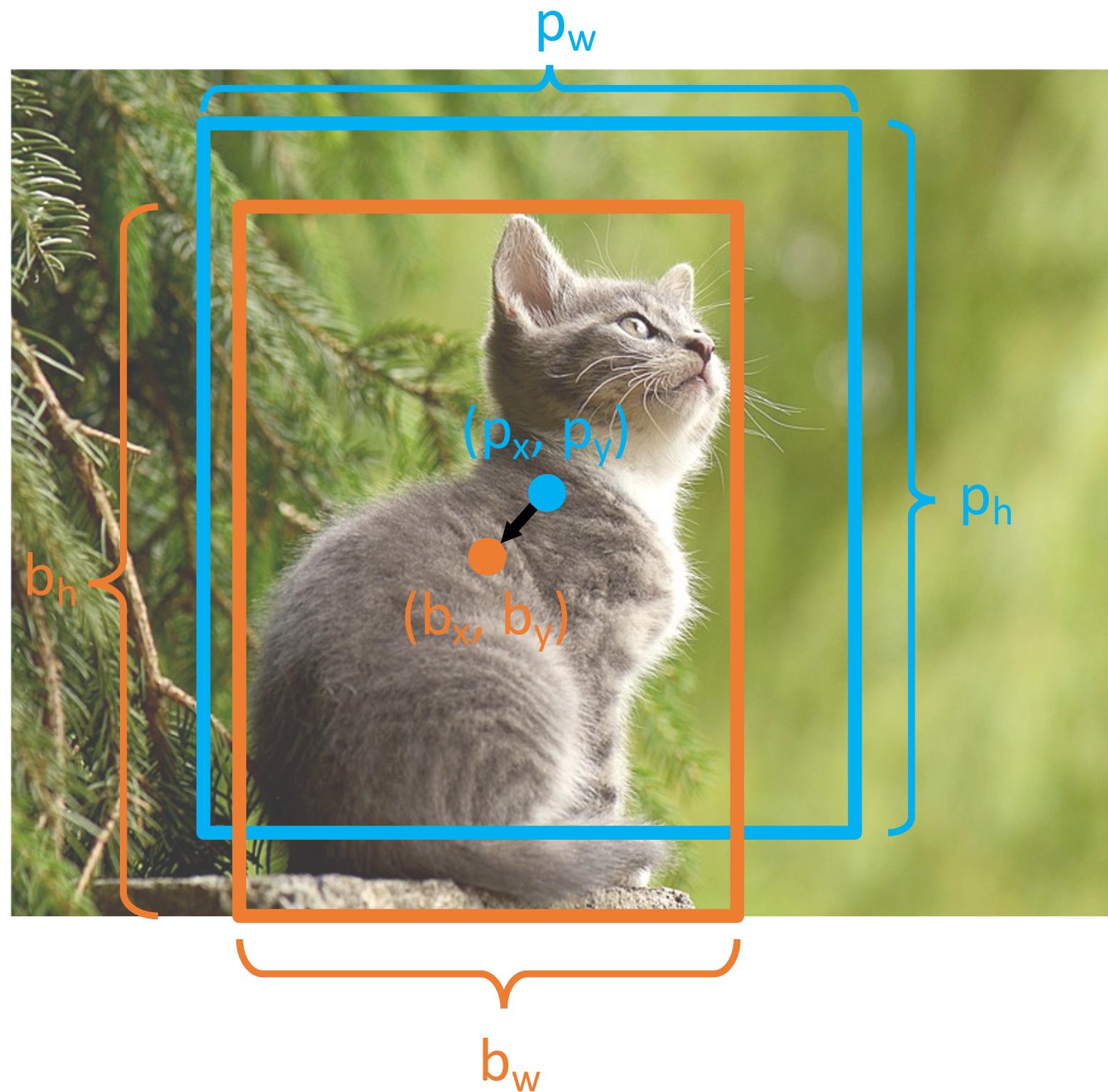
$$b_h = p_h \exp(t_h)$$

When transform is 0,
output = proposal

L2 regularization
encourages leaving
proposal unchanged

R-CNN: Box Regression

Consider a **region proposal** with center (p_x, p_y) , width p_w , height p_h



Model predicts a **transform** (t_x, t_y, t_w, t_h) to correct the region proposal

The **output box** is defined by:

$$b_x = p_x + p_w t_x$$

$$b_y = p_y + p_h t_y$$

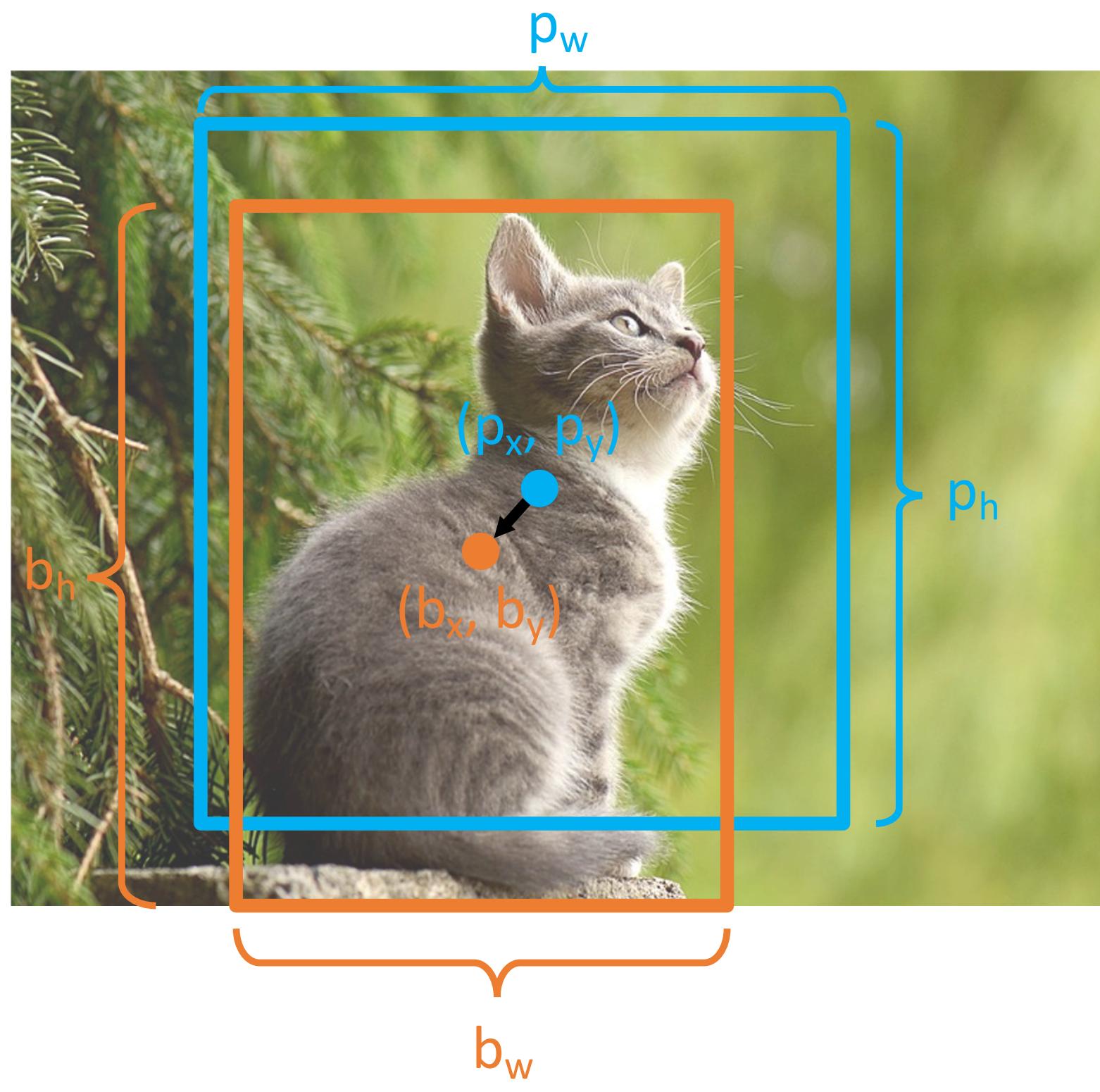
$$b_w = p_w \exp(t_w)$$

$$b_h = p_h \exp(t_h)$$

Scale / Translation invariance:
Transform encodes *relative* difference between proposal and output; important since CNN doesn't see absolute size or position after cropping

R-CNN: Box Regression

Consider a **region proposal** with center (p_x, p_y) , width p_w , height p_h



Model predicts a **transform** (t_x, t_y, t_w, t_h) to correct the region proposal

The **output box** is defined by

$$\begin{aligned} b_x &= p_x + p_w t_x \\ b_y &= p_y + p_h t_y \\ b_w &= p_w \exp(t_w) \\ b_h &= p_h \exp(t_h) \end{aligned}$$

Given **proposal** and **target output**, we can solve for the **transform** the network should output:

$$\begin{aligned} t_x &= (b_x - p_x)/p_w \\ t_y &= (b_y - p_y)/p_h \\ t_w &= \log(b_w/p_w) \\ t_h &= \log(b_h/p_h) \end{aligned}$$

R-CNN: Training

Input Image



Ground Truth

R-CNN: Training

Input Image

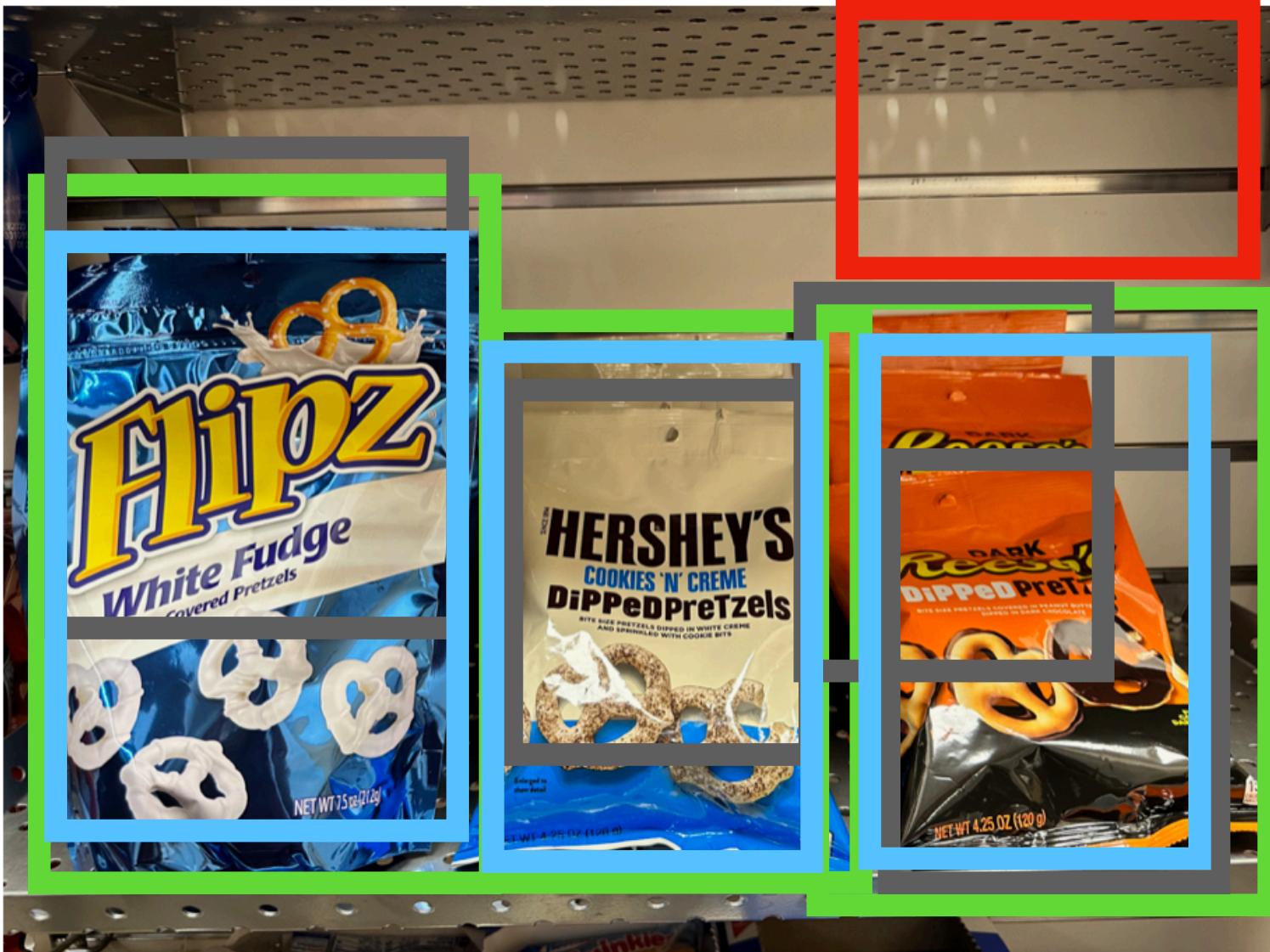


Ground Truth

Region Proposals

R-CNN: Training

Input Image



Ground Truth

Positive

Neutral

Negative

R-CNN: Training

Input Image



Categorize each region proposal as **positive**, **negative** or neutral based on overlap with the Ground truth boxes:

Positive: > 0.5 IoU with a GT box

Negative: < 0.3 IoU with all GT boxes

Neutral: between 0.3 and 0.5 IoU with GT boxes

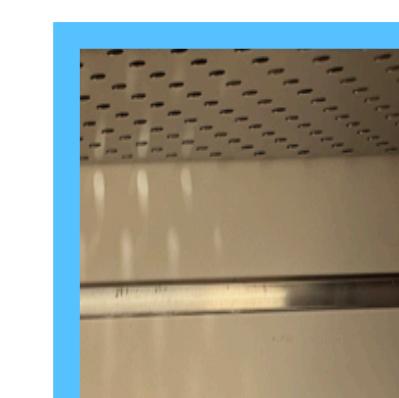
Ground Truth	Positive
Neutral	Negative

R-CNN: Training

Input Image



Ground Truth	Positive
Neutral	Negative



Crop pixels from each positive and negative proposal, resize to 224 x 224

Run each region through CNN
Positive regions: predict class and transform
Negative regions: just predict class

R-CNN: Training

Input Image



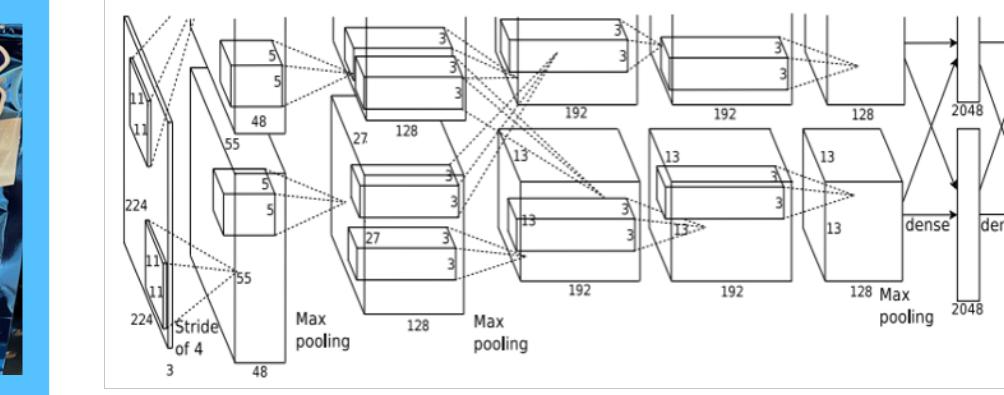
Ground Truth

Positive

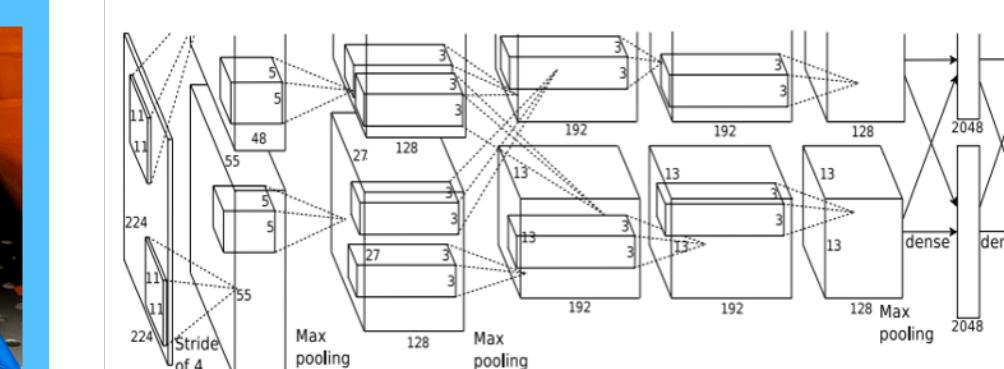
Neutral

Negative

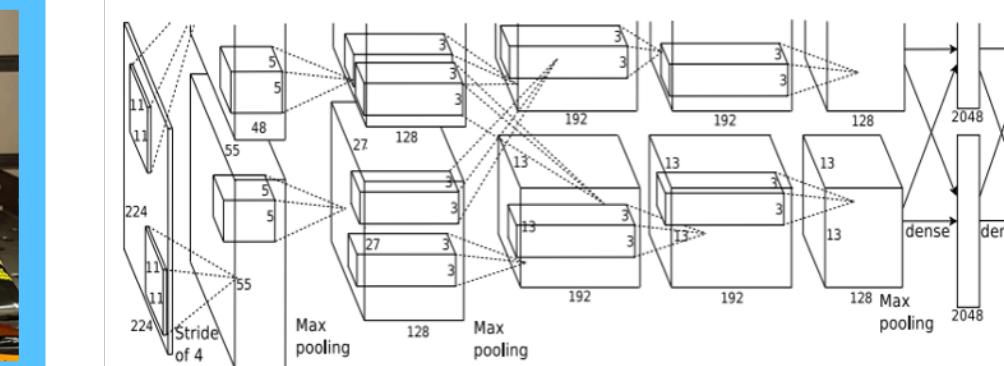
Run each region through CNN
 Positive regions: predict class and transform
 Negative regions: just predict class



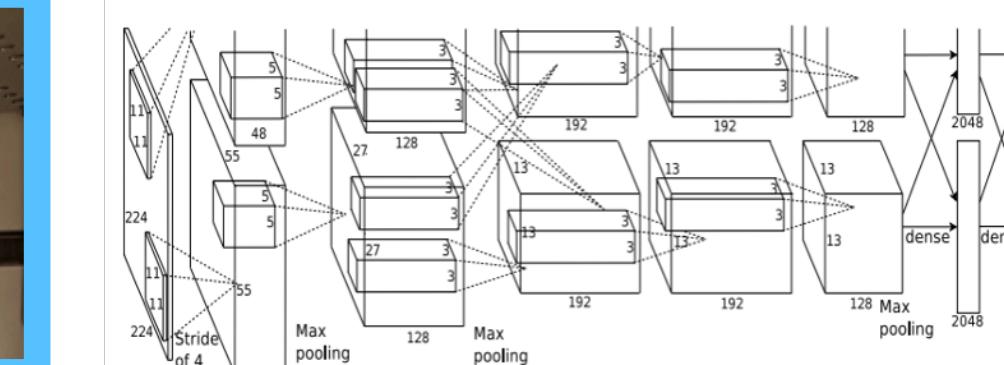
Class target: Flipz
 Box target: —→



Class target: Hershey's
 Box target: —→



Class target: Reese's
 Box target: —→



Class target: Background
 Box target: None

R-CNN: Test time

Input Image



Region Proposals

Run proposal method:

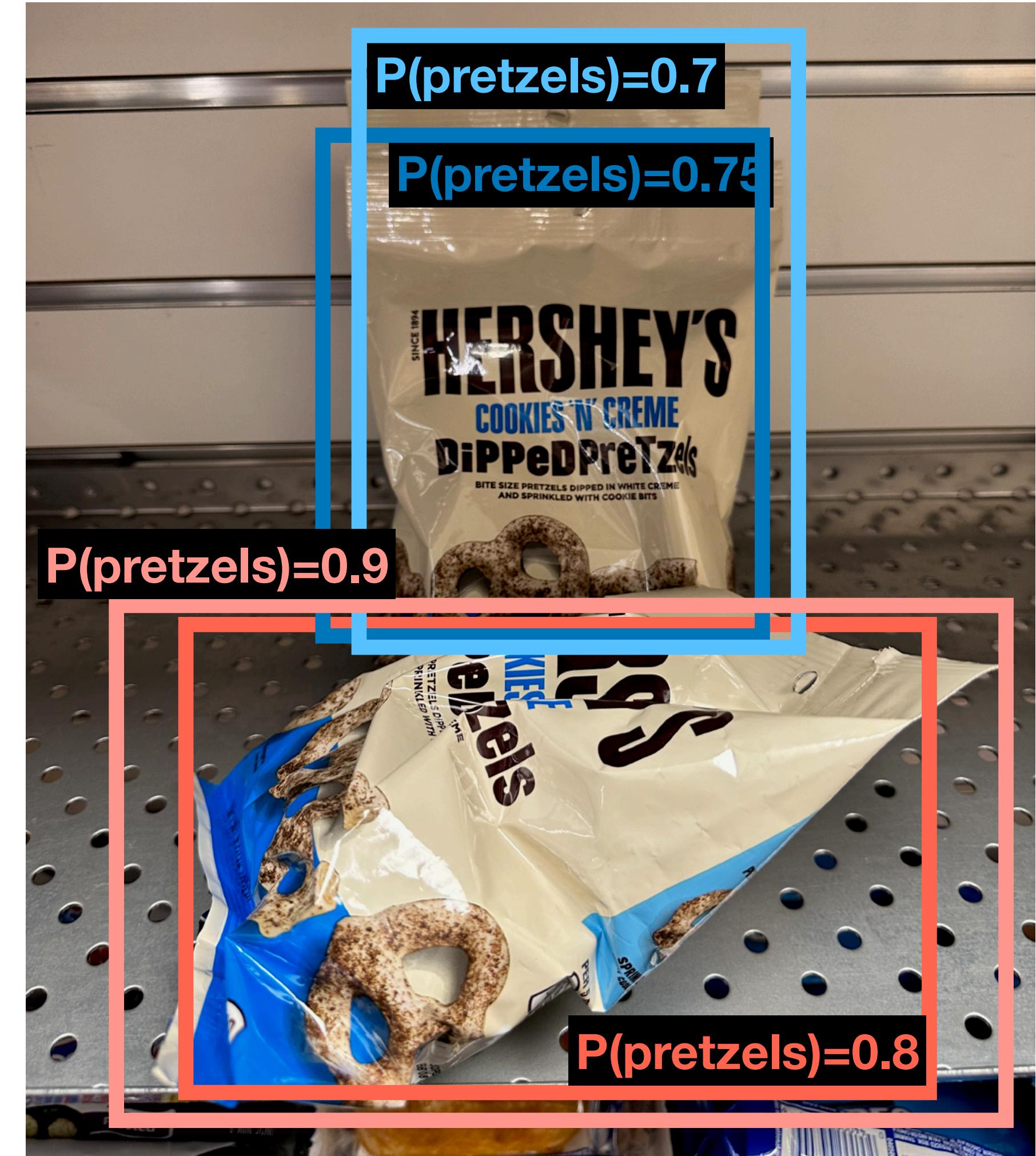
1. Run CNN on each proposal to get class scores, transforms
2. Threshold class scores to get a set of detections

2 Problems:

1. CNN often outputs overlapping boxes
2. How to set thresholds?

Overlapping Boxes

Problem: Object detectors often output many overlapping detections

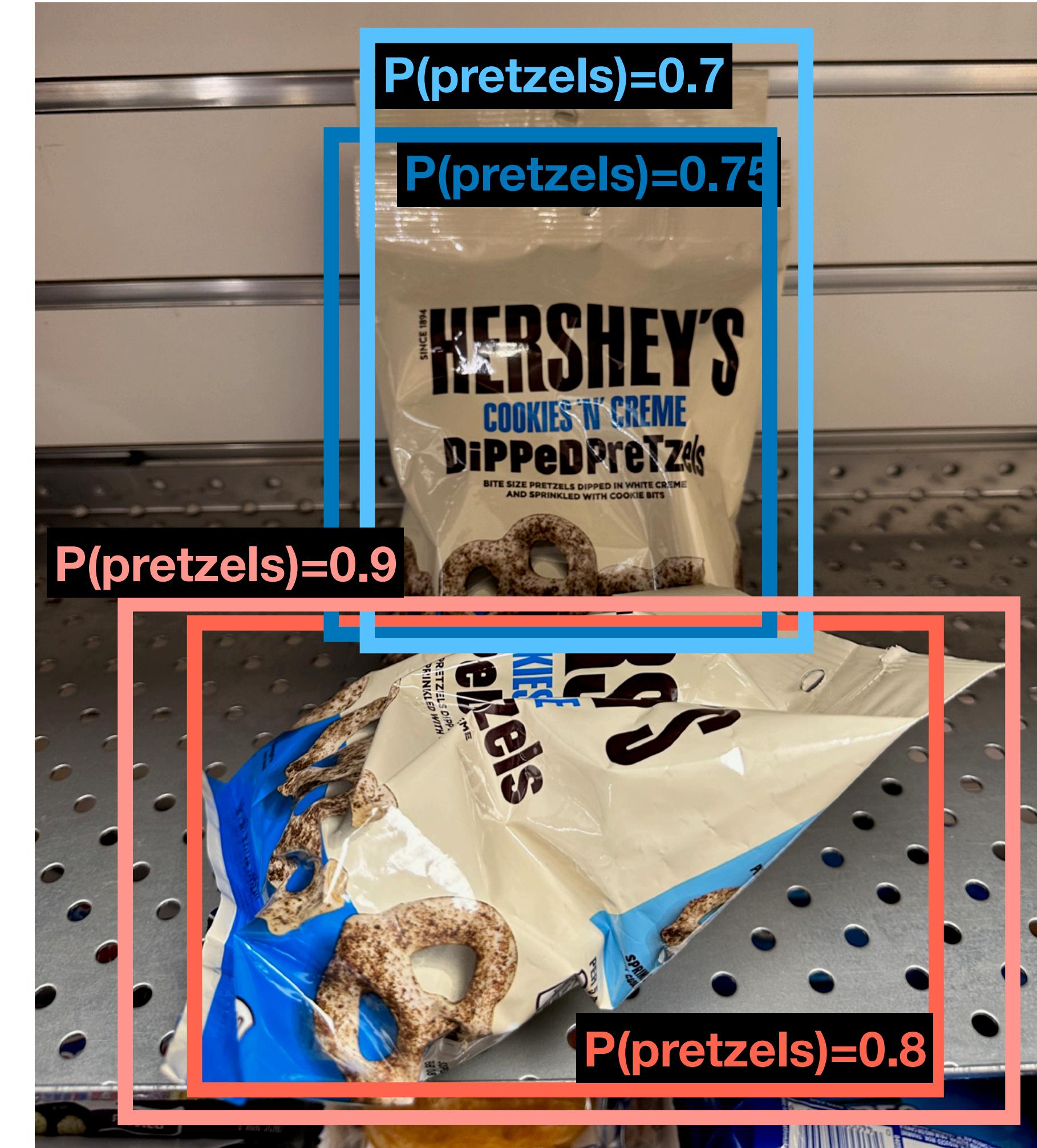


Overlapping Boxes: Non-Max Suppression (NMS)

Problem: Object detectors often output many overlapping detections

Solution: Post-process raw detections using Non-Max Suppression (NMS)

1. Select next highest-scoring box
2. Eliminate lower-scoring boxes with $\text{IoU} >$ threshold (e.g. 0.7)
3. If any boxes remain, GOTO 1



Overlapping Boxes: Non-Max Suppression (NMS)

Problem: Object detectors often output many overlapping detections

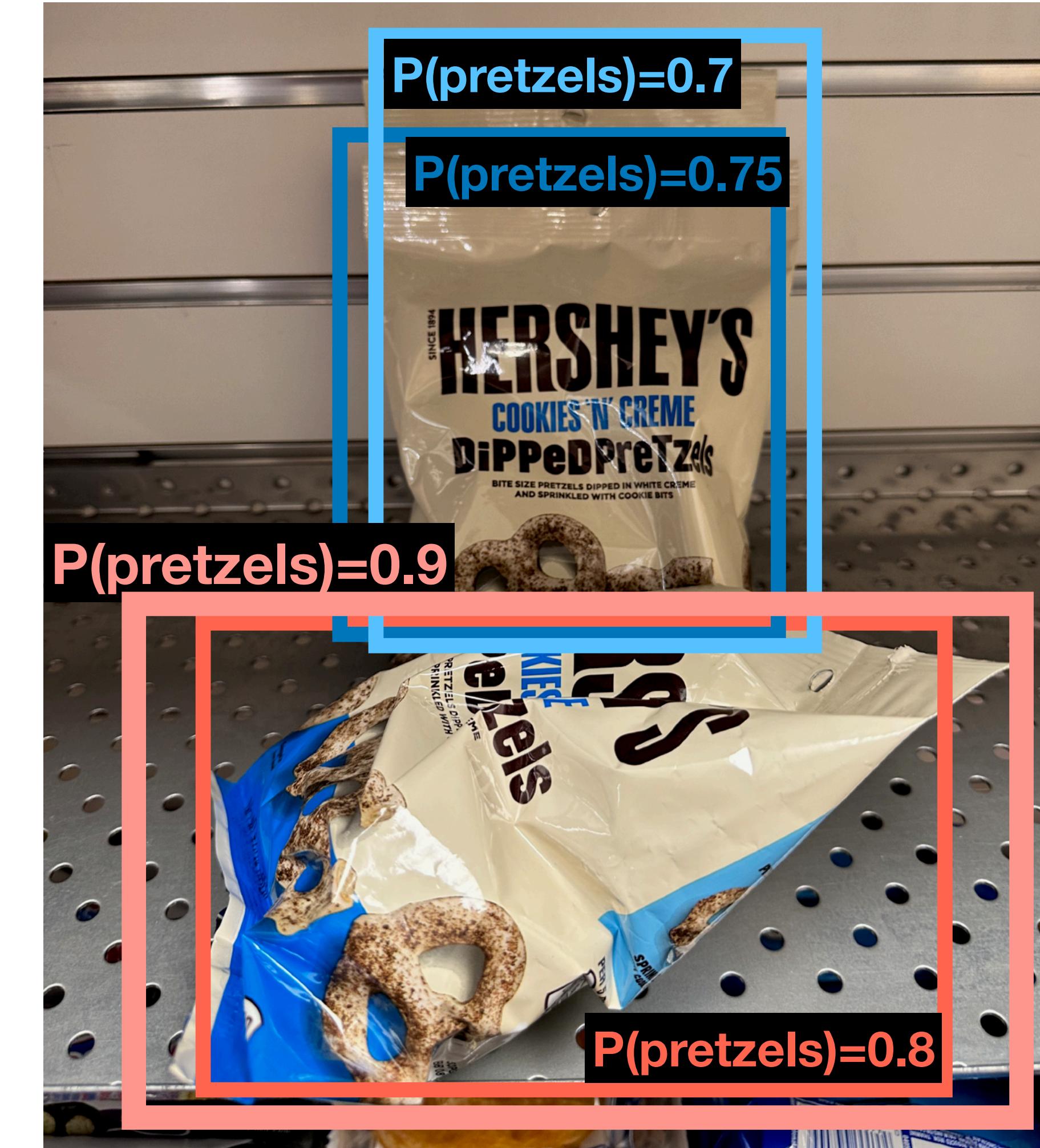
Solution: Post-process raw detections using Non-Max Suppression (NMS)

1. Select next highest-scoring box
2. Eliminate lower-scoring boxes with $\text{IoU} >$ threshold (e.g. 0.7)
3. If any boxes remain, GOTO 1

$$\text{IoU}(\text{pink}, \text{red}) = 0.8$$

$$\text{IoU}(\text{pink}, \text{blue}) = 0.03$$

$$\text{IoU}(\text{pink}, \text{light blue}) = 0.05$$



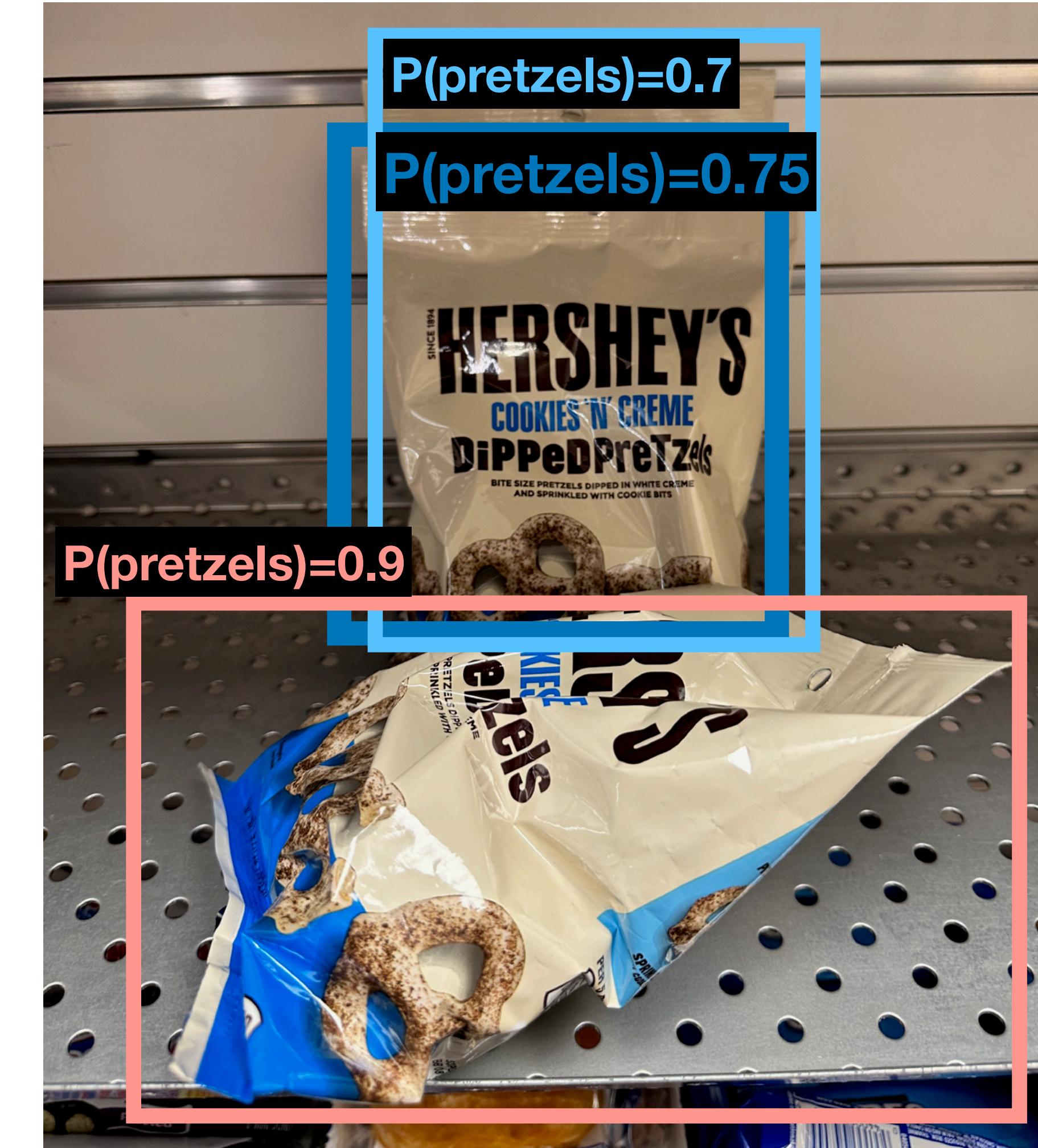
Overlapping Boxes: Non-Max Suppression (NMS)

Problem: Object detectors often output many overlapping detections

Solution: Post-process raw detections using Non-Max Suppression (NMS)

1. Select next highest-scoring box
2. Eliminate lower-scoring boxes with $\text{IoU} >$ threshold (e.g. 0.7)
3. If any boxes remain, GOTO 1

$$\text{IoU}(\square, \square) = 0.85$$

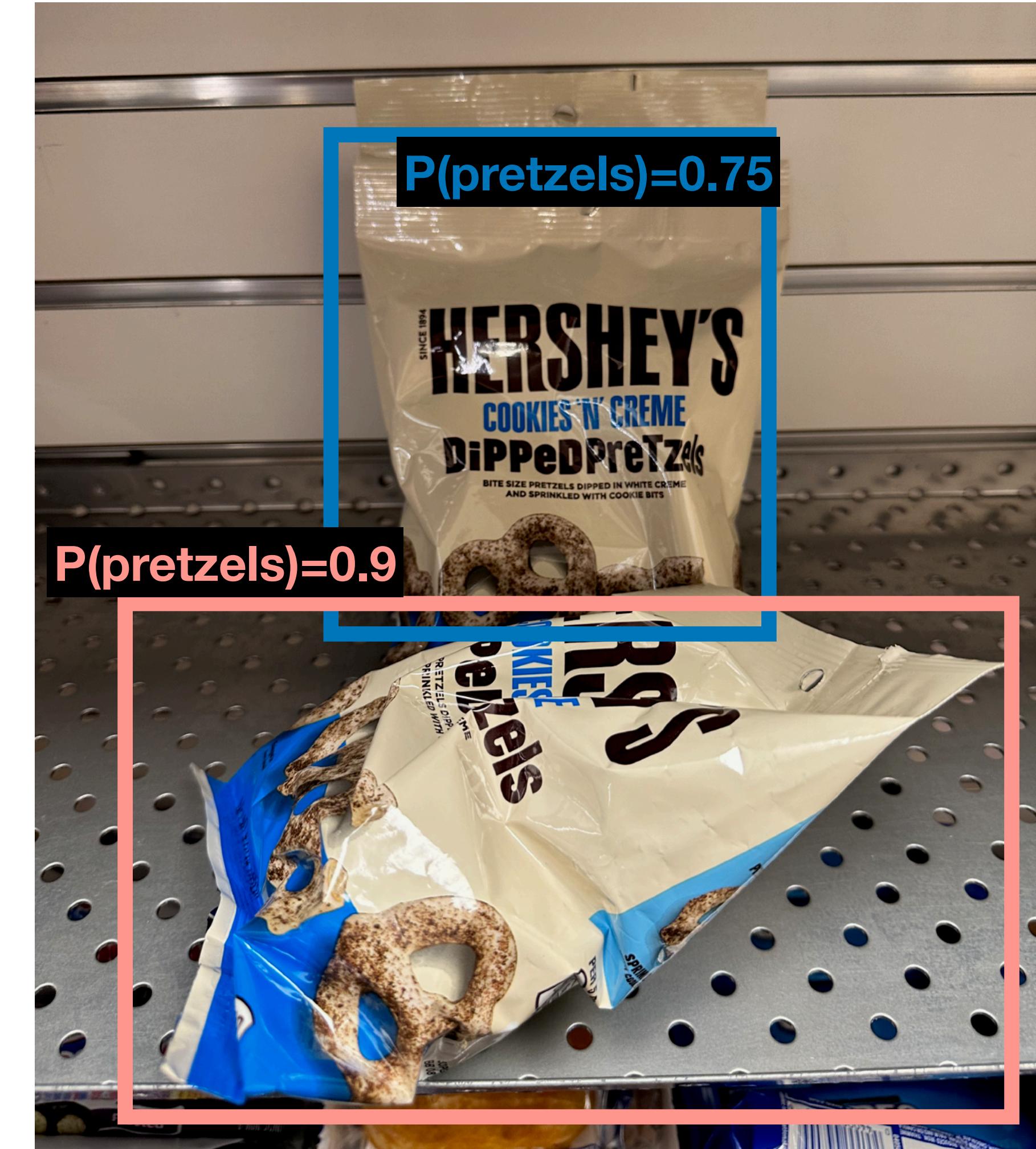


Overlapping Boxes: Non-Max Suppression (NMS)

Problem: Object detectors often output many overlapping detections

Solution: Post-process raw detections using Non-Max Suppression (NMS)

1. Select next highest-scoring box
2. Eliminate lower-scoring boxes with $\text{IoU} >$ threshold (e.g. 0.7)
3. If any boxes remain, GOTO 1



Overlapping Boxes: Non-Max Suppression (NMS)

Problem: Object detectors often output many overlapping detections

Solution: Post-process raw detections using Non-Max Suppression (NMS)

1. Select next highest-scoring box
2. Eliminate lower-scoring boxes with $\text{IoU} >$ threshold (e.g. 0.7)
3. If any boxes remain, GOTO 1

Problem: NMS may eliminate “good” boxes when objects are highly overlapping... no good solution



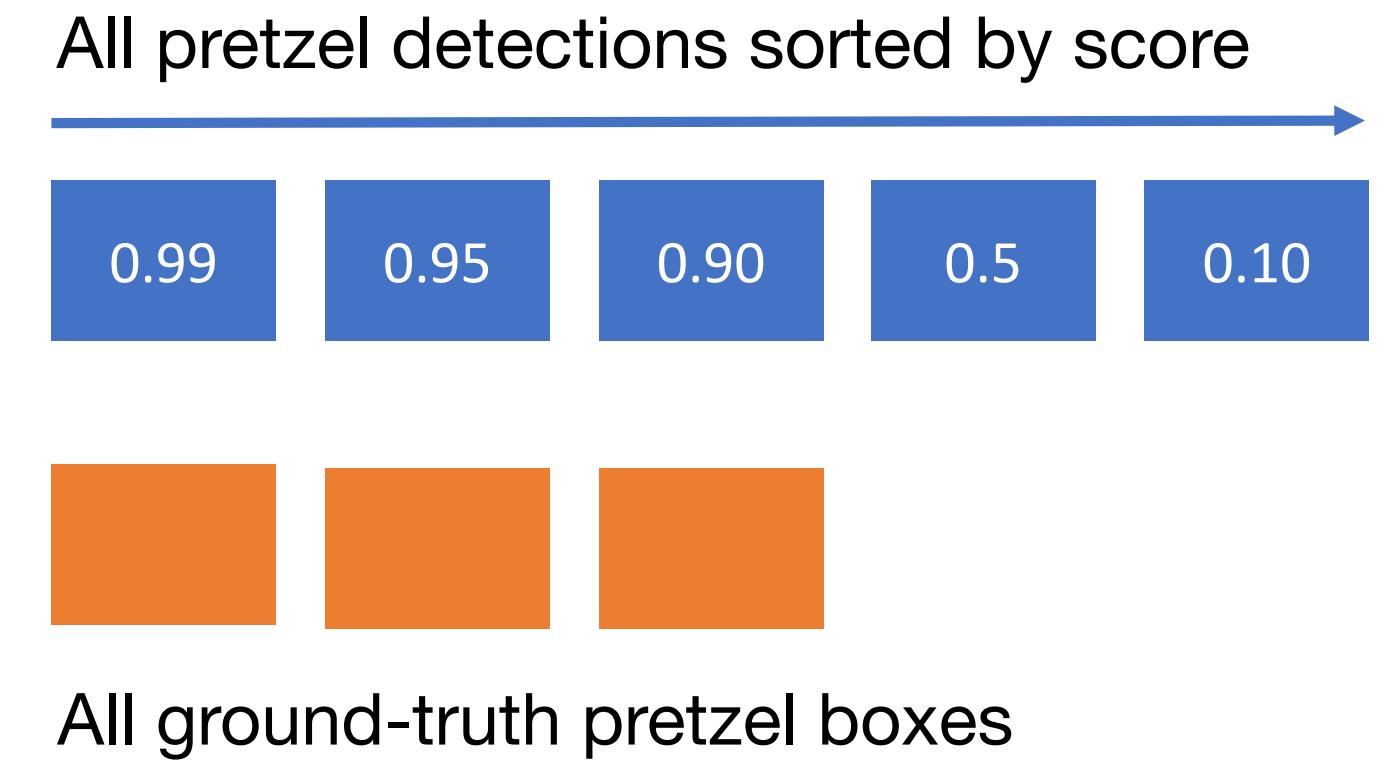
[Crowd image](#) is free for commercial use under the [Pixabay license](#)

Evaluating Object Detectors: Mean Average Precision (mAP)

1. Run object detector on all test images (with NMS)
2. For each category, compute Average Precision
(AP) = area under Precision vs Recall Curve

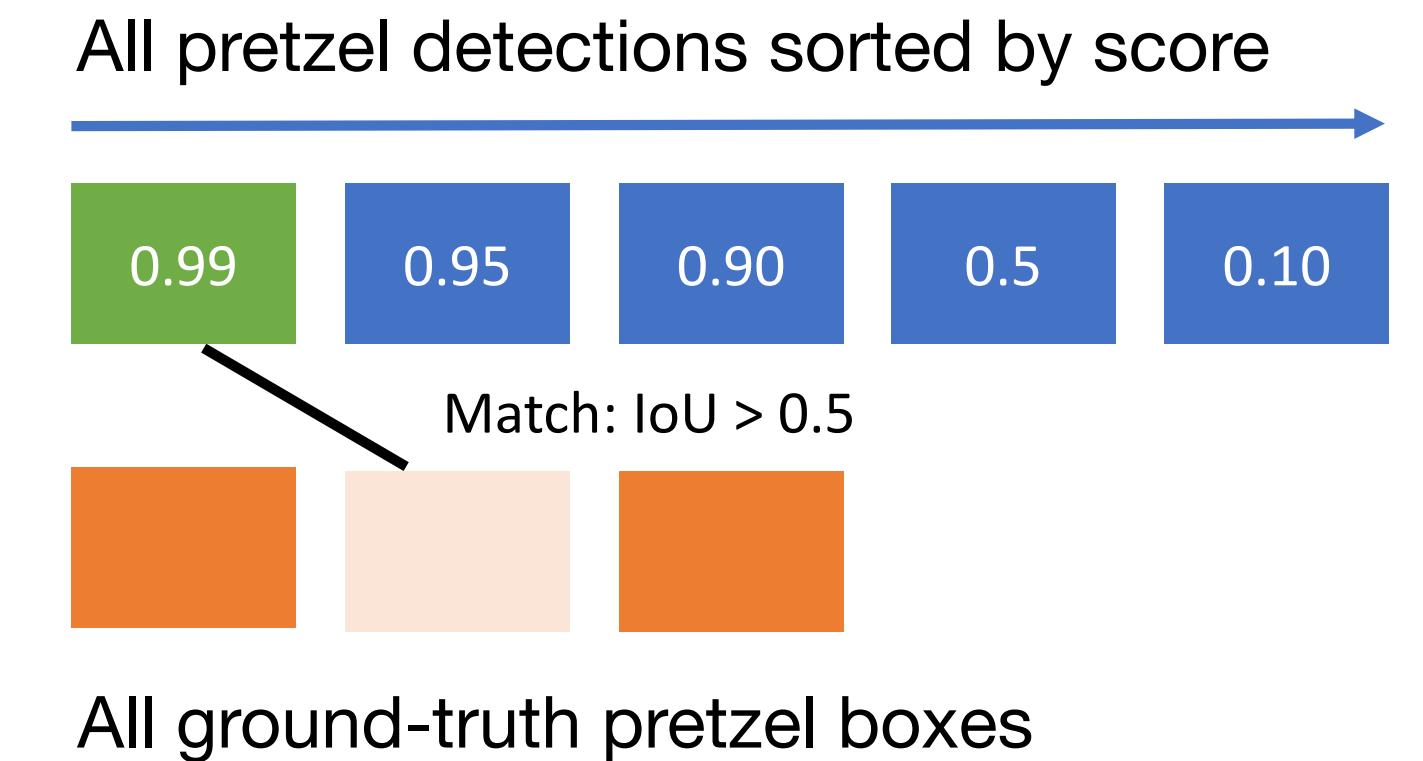
Evaluating Object Detectors: Mean Average Precision (mAP)

1. Run object detector on all test images (with NMS)
2. For each category, compute Average Precision
(AP) = area under Precision vs Recall Curve
 1. For each detection (highest score to lowest score)



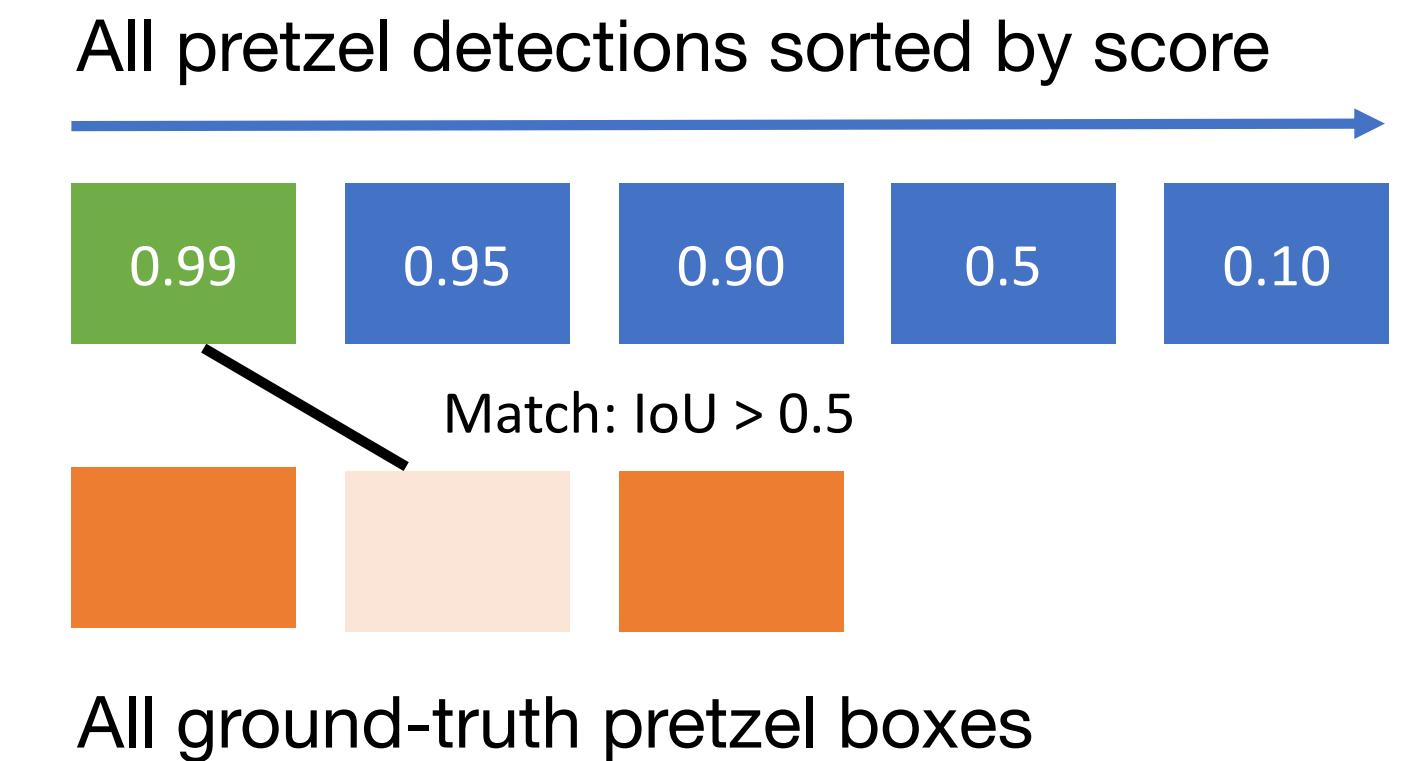
Evaluating Object Detectors: Mean Average Precision (mAP)

1. Run object detector on all test images (with NMS)
2. For each category, compute Average Precision
(AP) = area under Precision vs Recall Curve
 1. For each detection (highest score to lowest score)
 1. If it matches some GT box with $\text{IoU} > 0.5$, mark it as positive and eliminate the GT
 2. Otherwise mark it as negative

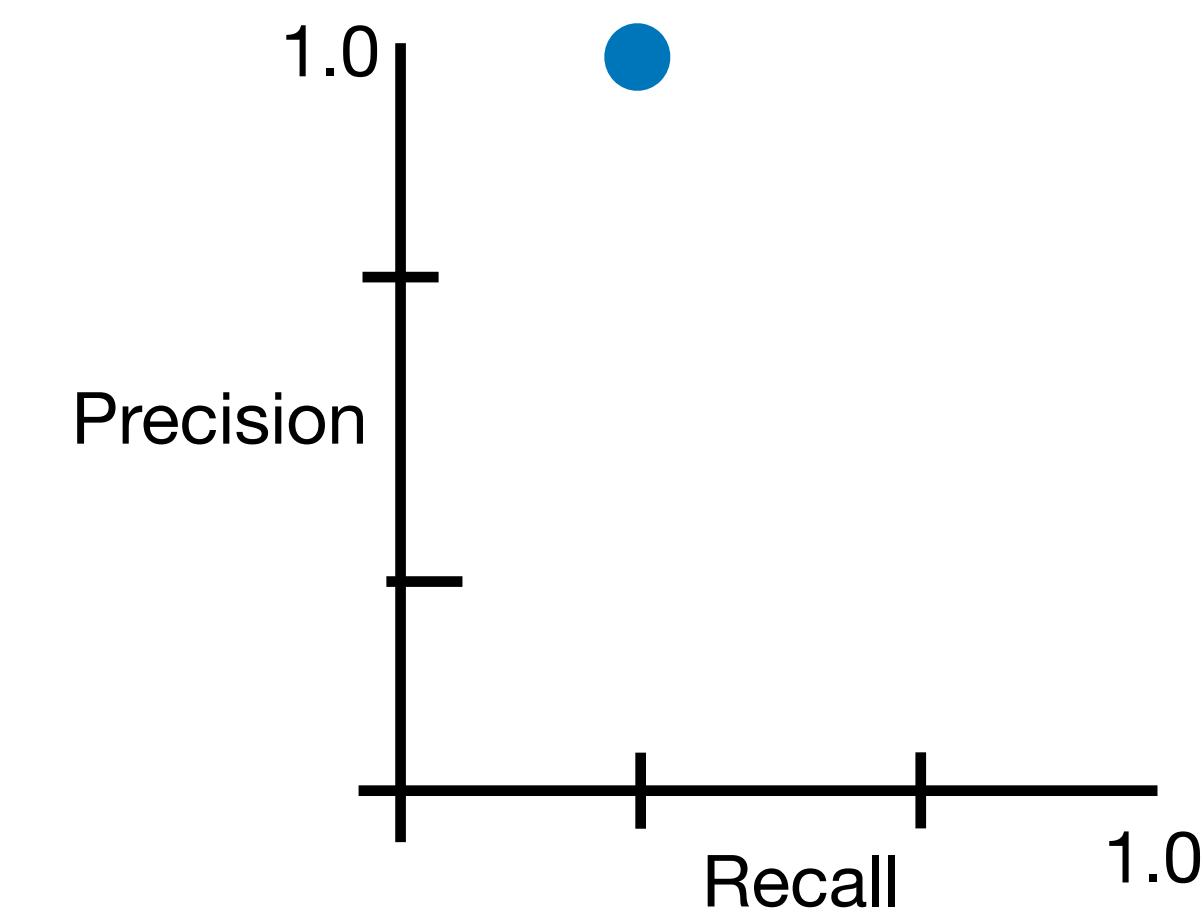


Evaluating Object Detectors: Mean Average Precision (mAP)

1. Run object detector on all test images (with NMS)
2. For each category, compute Average Precision
(AP) = area under Precision vs Recall Curve
 1. For each detection (highest score to lowest score)
 1. If it matches some GT box with $\text{IoU} > 0.5$, mark it as positive and eliminate the GT
 2. Otherwise mark it as negative
 3. Plot a point on PR curve

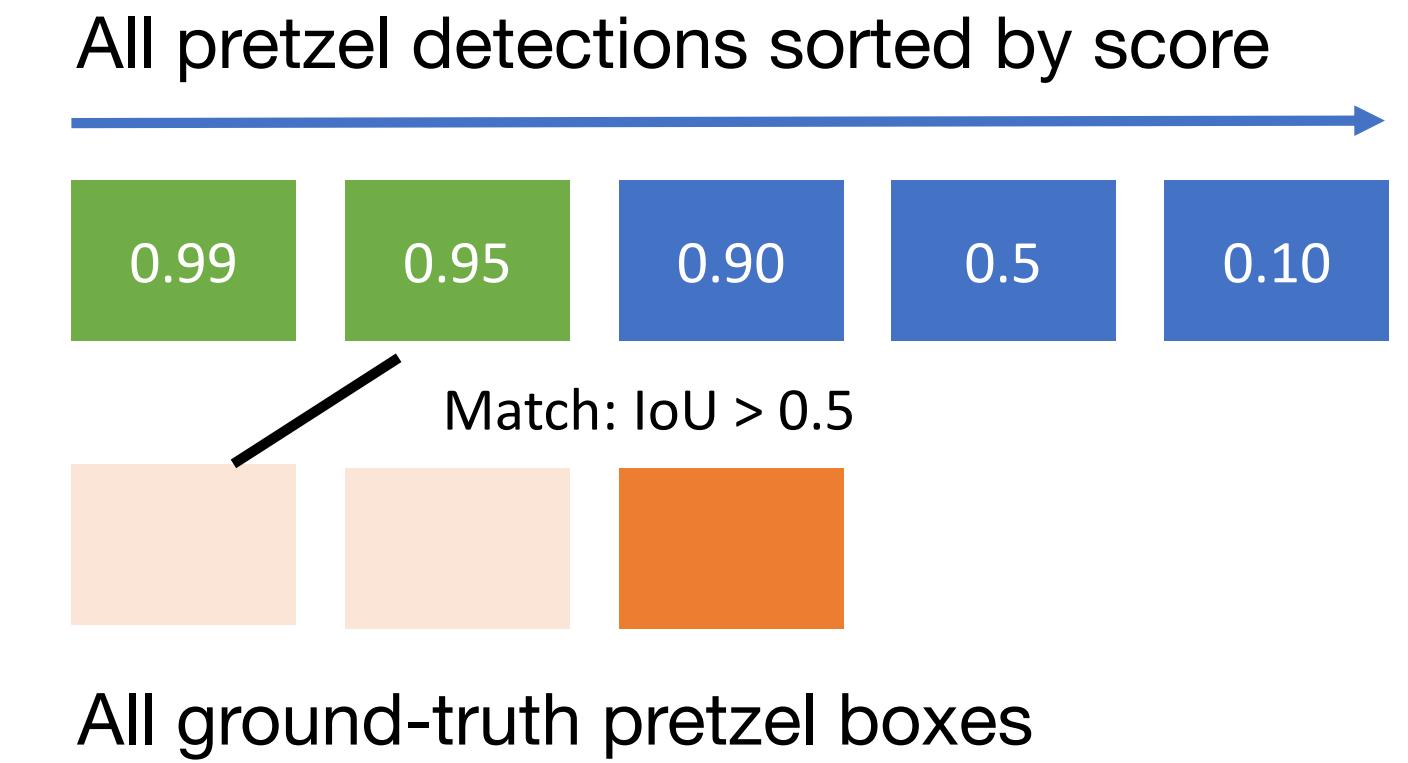


Precision = 1/1 = 1.0
Recall = 1/3 = 0.33

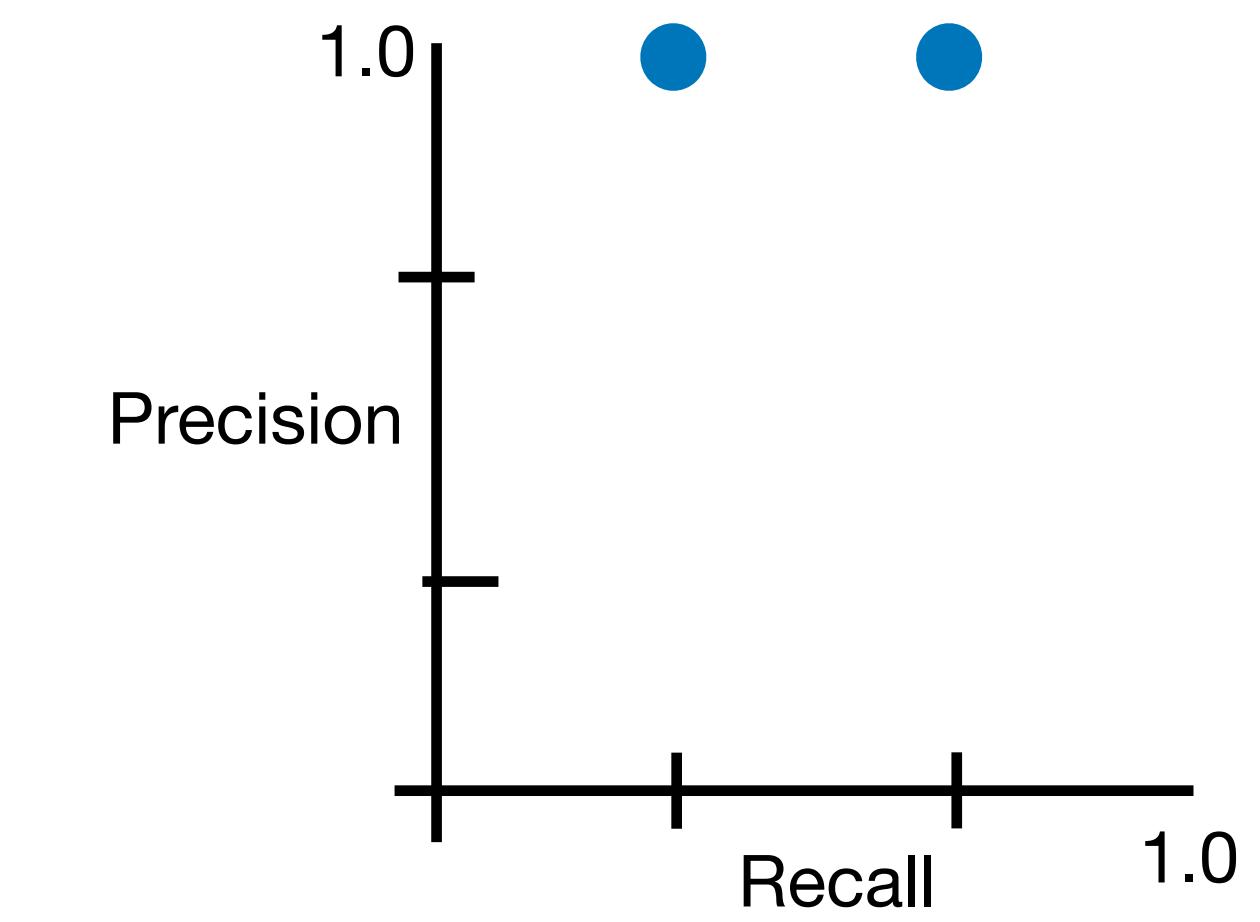


Evaluating Object Detectors: Mean Average Precision (mAP)

1. Run object detector on all test images (with NMS)
2. For each category, compute Average Precision
(AP) = area under Precision vs Recall Curve
 1. For each detection (highest score to lowest score)
 1. If it matches some GT box with $\text{IoU} > 0.5$, mark it as positive and eliminate the GT
 2. Otherwise mark it as negative
 3. Plot a point on PR curve

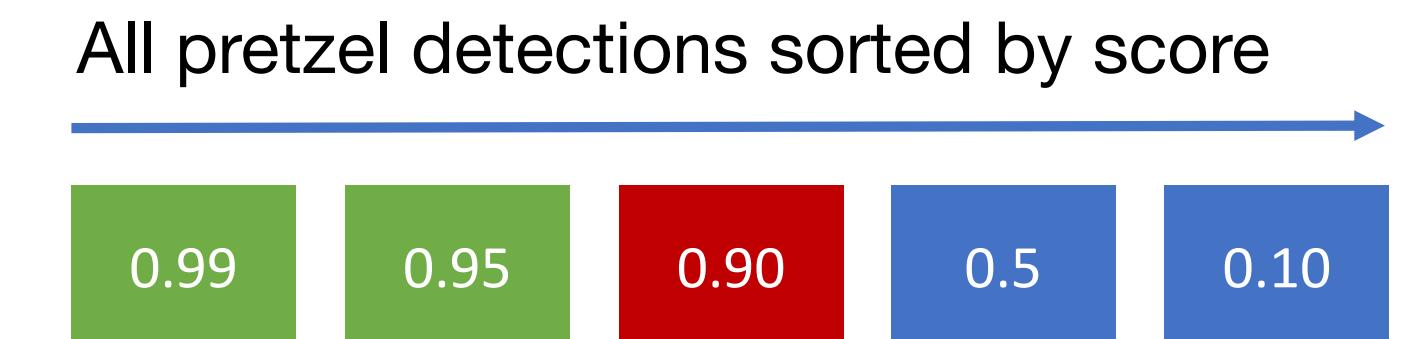


Precision = $2/2 = 1.0$
Recall = $2/3 = 0.67$



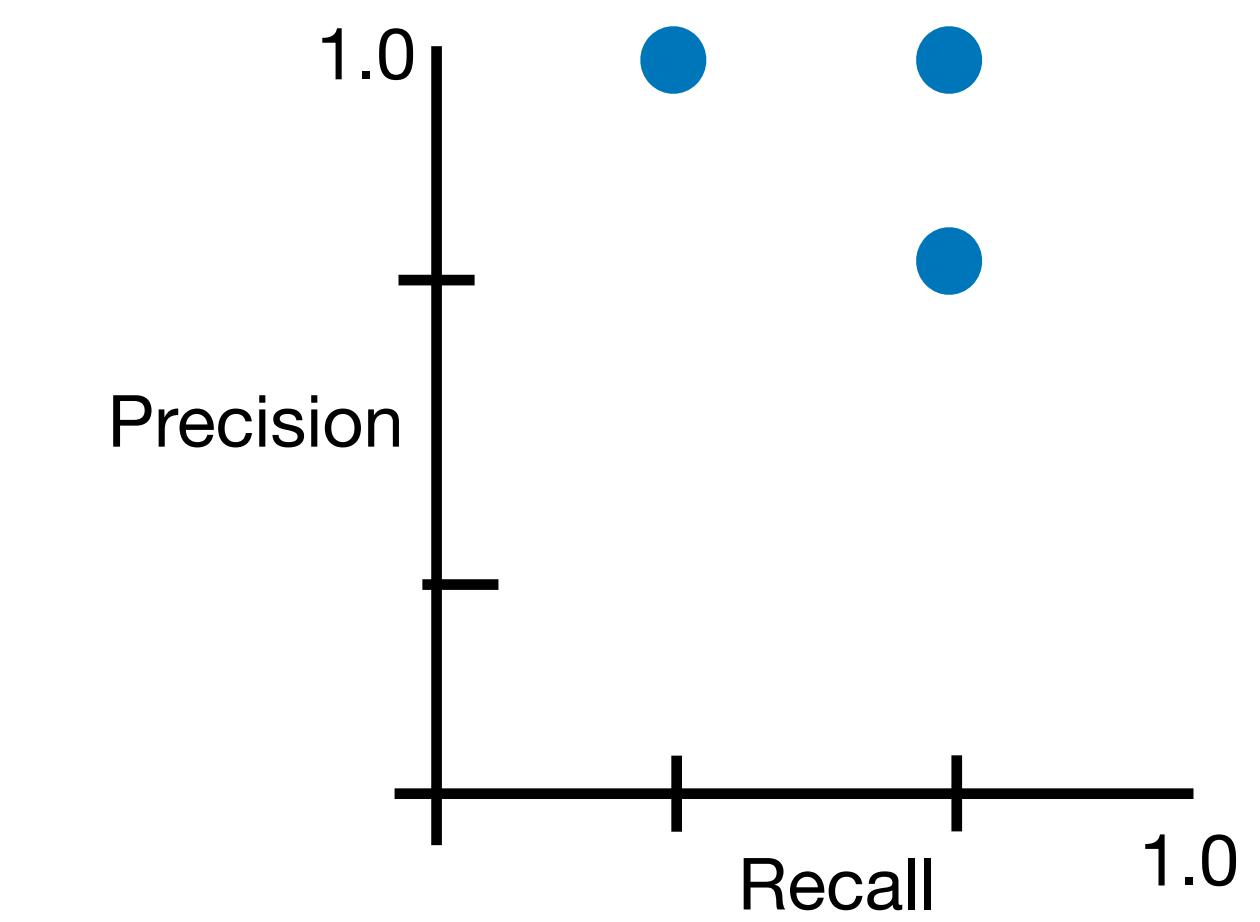
Evaluating Object Detectors: Mean Average Precision (mAP)

1. Run object detector on all test images (with NMS)
2. For each category, compute Average Precision
(AP) = area under Precision vs Recall Curve
 1. For each detection (highest score to lowest score)
 1. If it matches some GT box with $\text{IoU} > 0.5$, mark it as positive and eliminate the GT
 2. Otherwise mark it as negative
 3. Plot a point on PR curve



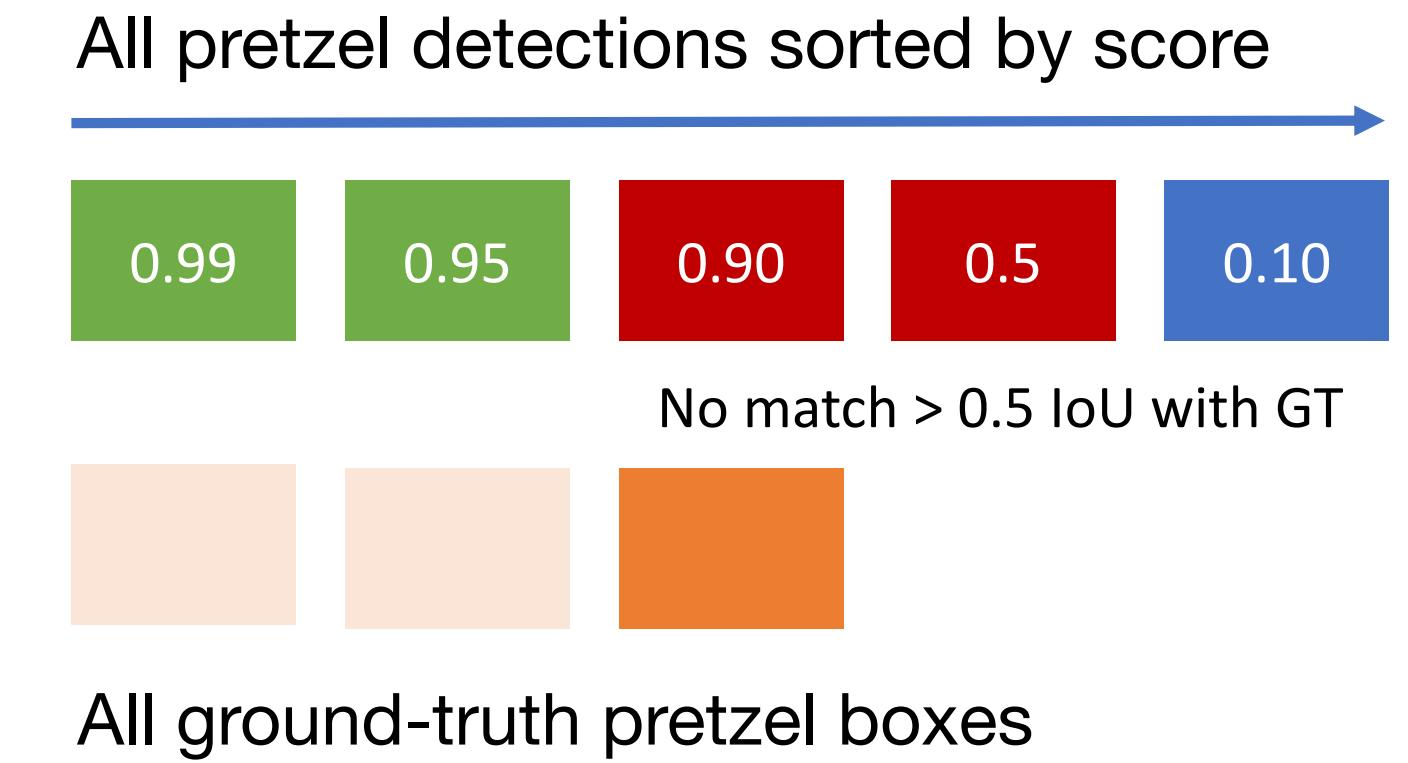
All ground-truth pretzel boxes

Precision = $2/3 = 0.67$
Recall = $2/3 = 0.67$

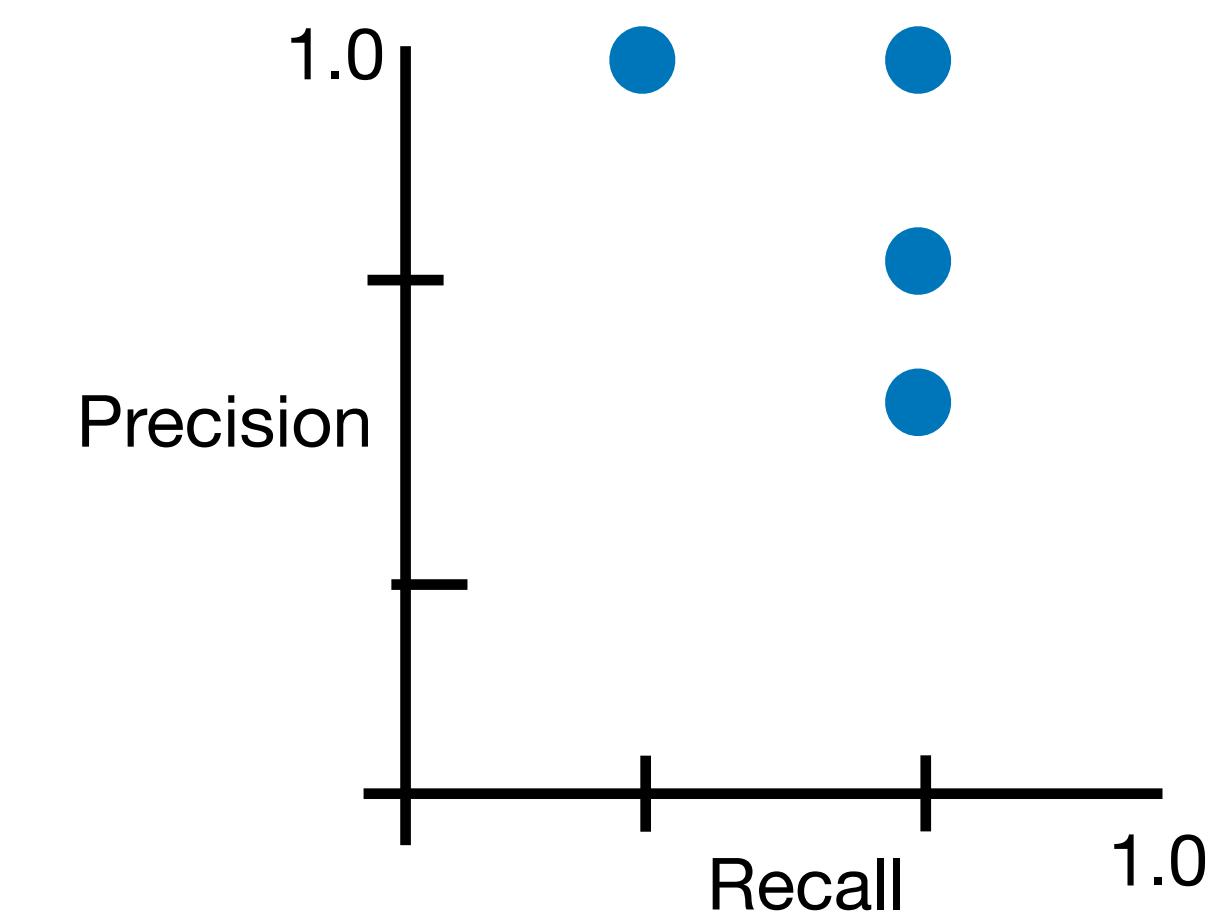


Evaluating Object Detectors: Mean Average Precision (mAP)

1. Run object detector on all test images (with NMS)
2. For each category, compute Average Precision
(AP) = area under Precision vs Recall Curve
 1. For each detection (highest score to lowest score)
 1. If it matches some GT box with $\text{IoU} > 0.5$, mark it as positive and eliminate the GT
 2. Otherwise mark it as negative
 3. Plot a point on PR curve

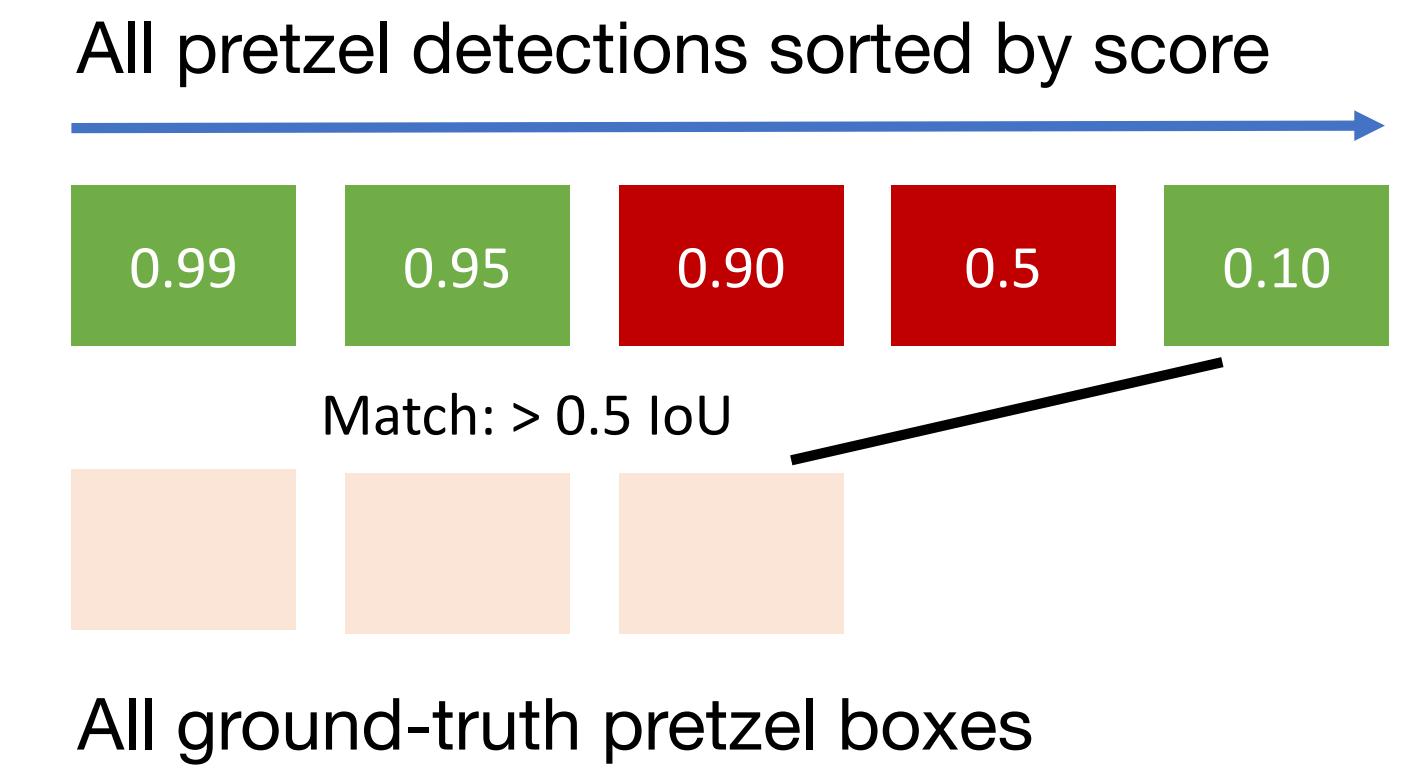


Precision = $2/4 = 0.5$
Recall = $2/3 = 0.67$

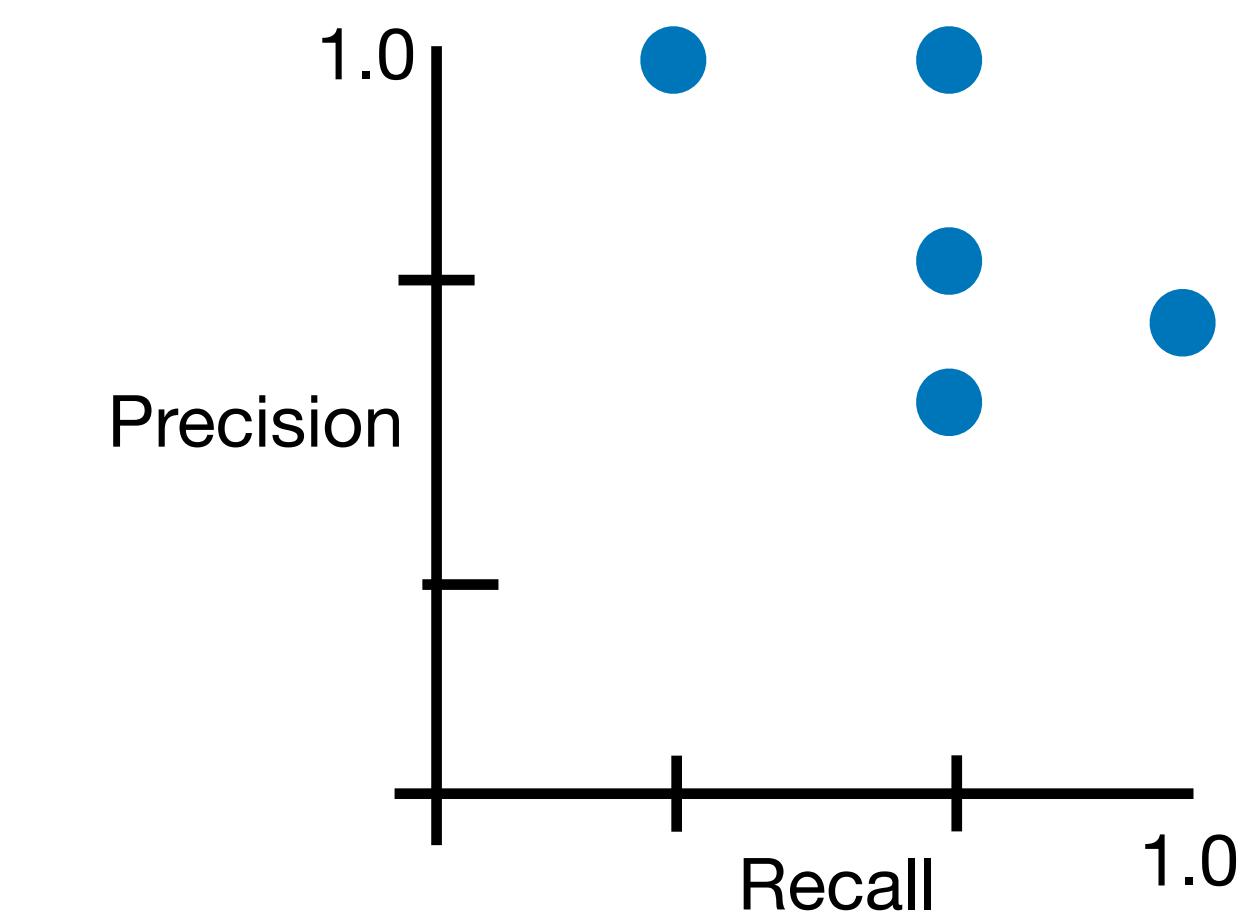


Evaluating Object Detectors: Mean Average Precision (mAP)

1. Run object detector on all test images (with NMS)
2. For each category, compute Average Precision
(AP) = area under Precision vs Recall Curve
 1. For each detection (highest score to lowest score)
 1. If it matches some GT box with $\text{IoU} > 0.5$, mark it as positive and eliminate the GT
 2. Otherwise mark it as negative
 3. Plot a point on PR curve

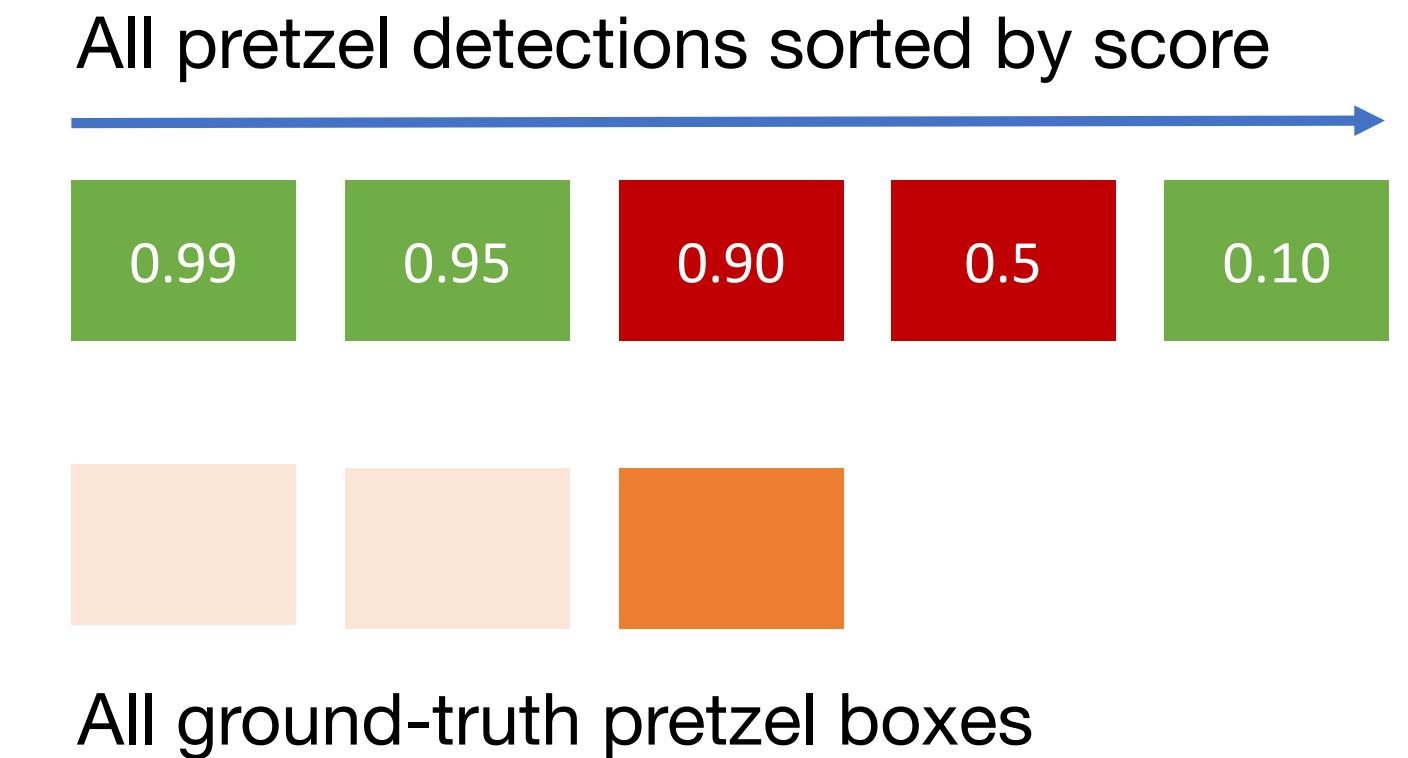


Precision = $3/5 = 0.6$
Recall = $3/3 = 1.0$

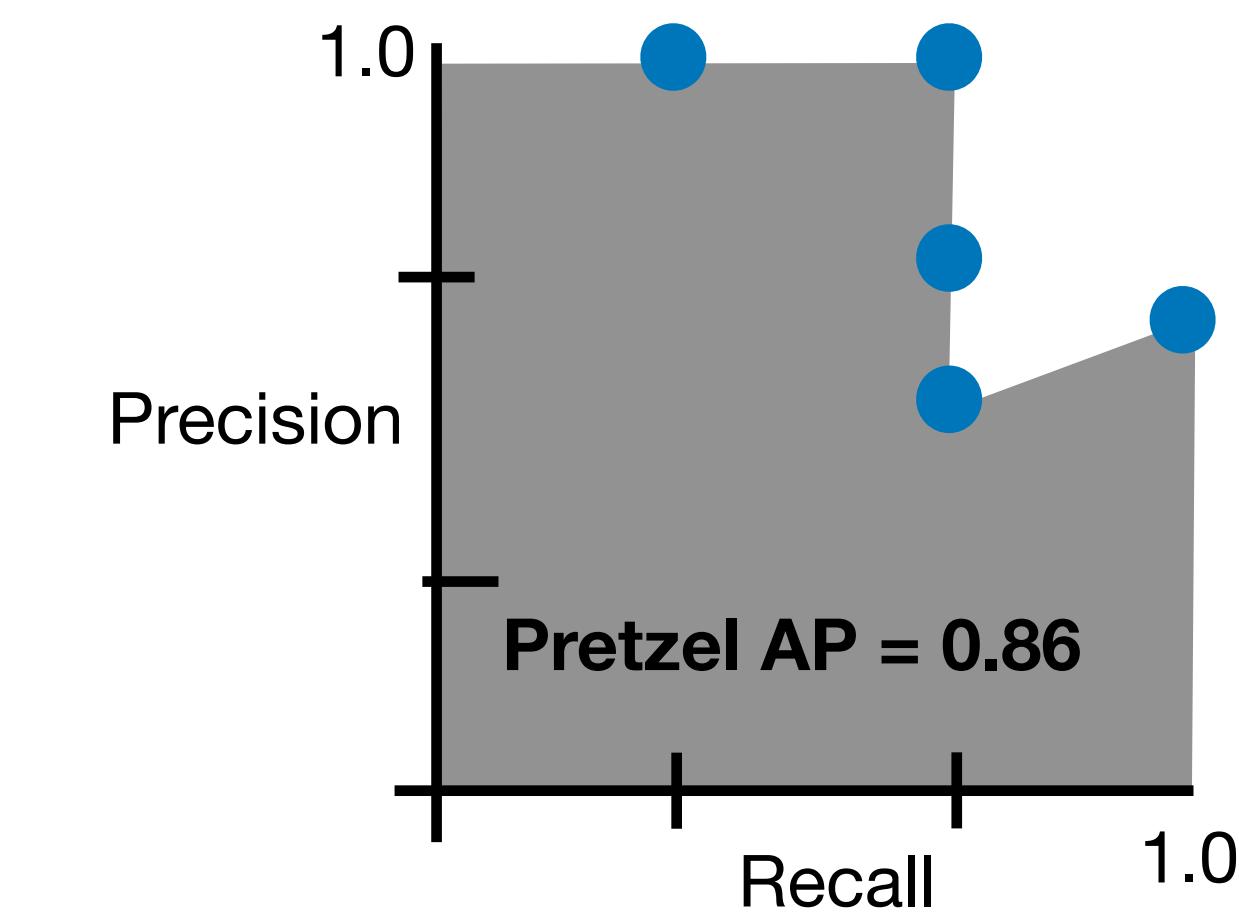


Evaluating Object Detectors: Mean Average Precision (mAP)

1. Run object detector on all test images (with NMS)
2. For each category, compute Average Precision
(AP) = area under Precision vs Recall Curve
 1. For each detection (highest score to lowest score)
 1. If it matches some GT box with $\text{IoU} > 0.5$, mark it as positive and eliminate the GT
 2. Otherwise mark it as negative
 3. Plot a point on PR curve
 2. Average Precision (AP) = area under PR curve



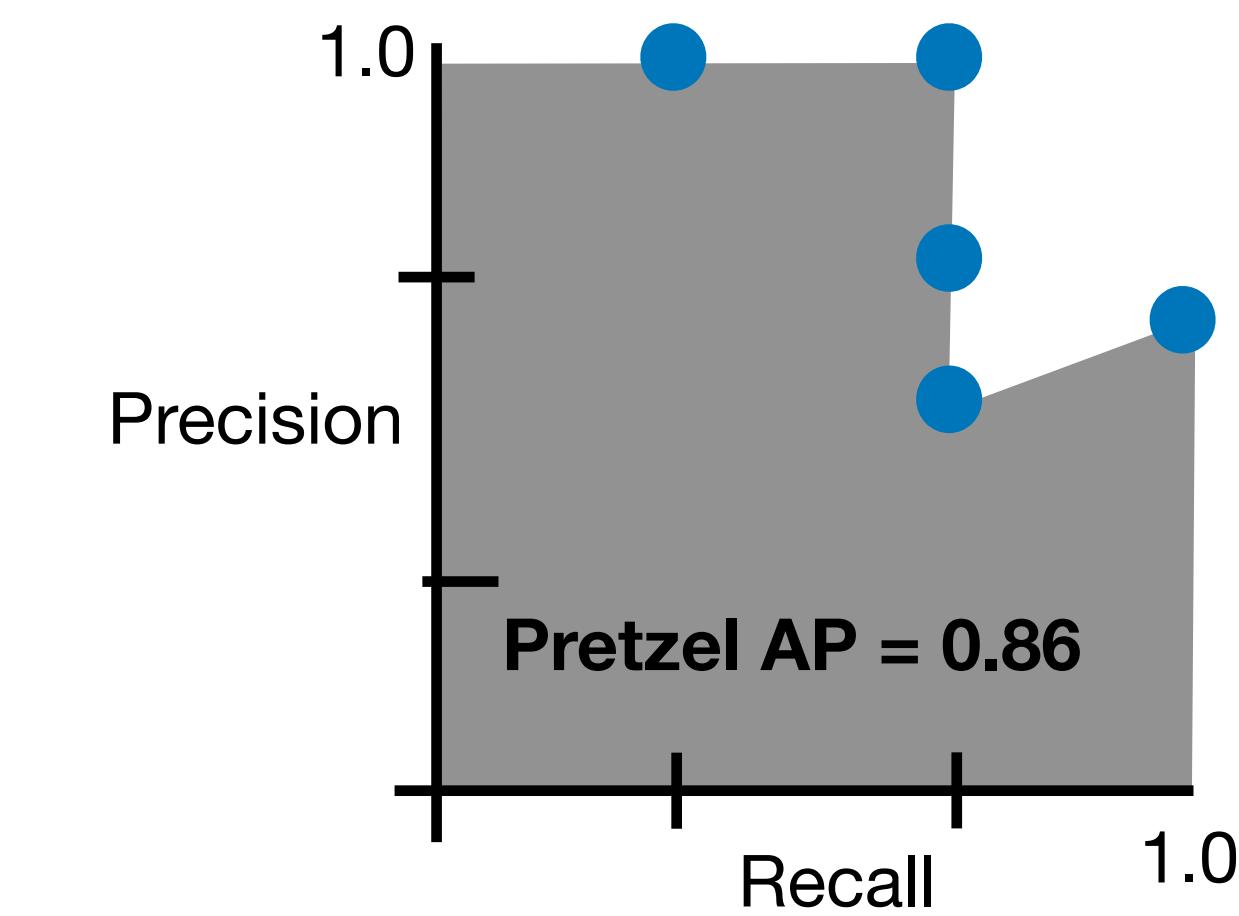
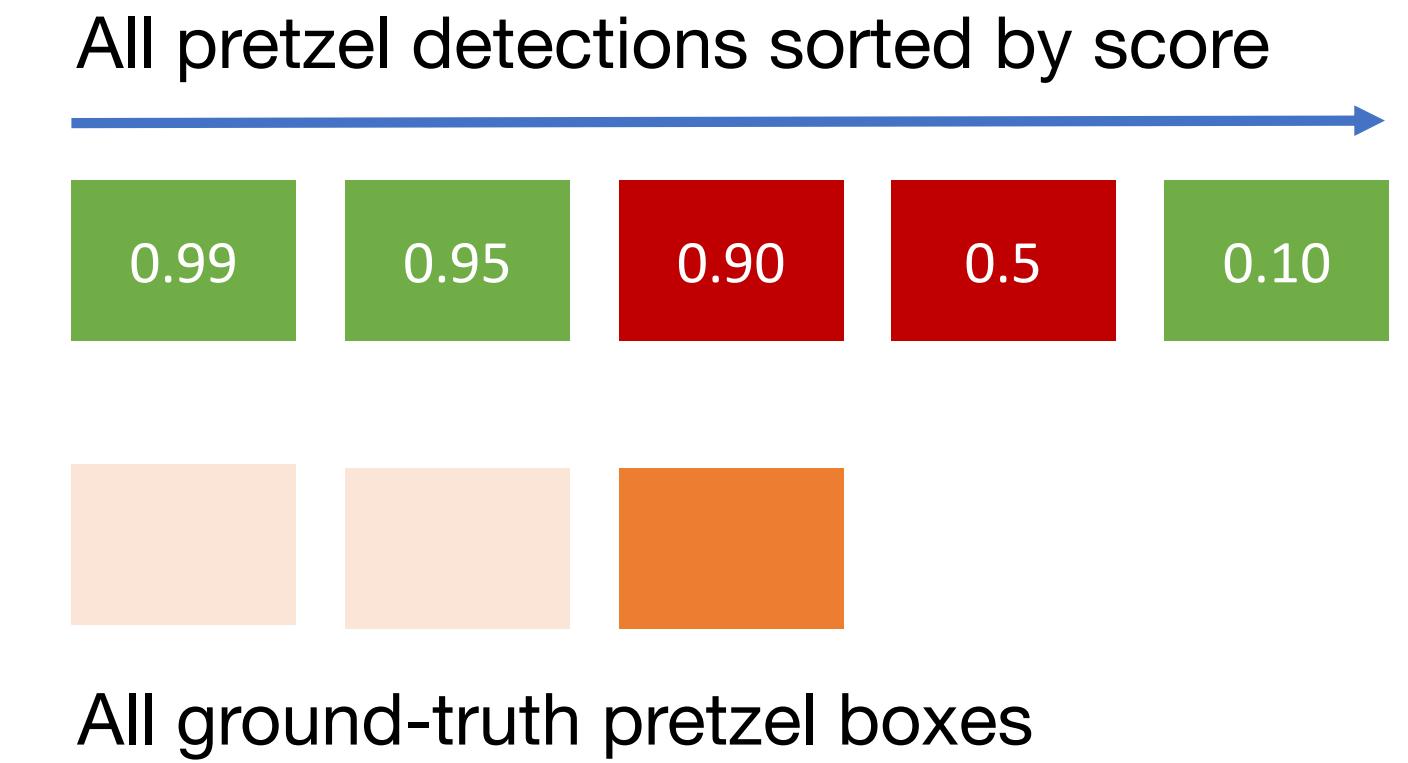
All ground-truth pretzel boxes



Evaluating Object Detectors: Mean Average Precision (mAP)

1. Run object detector on all test images (with NMS)
2. For each category, compute Average Precision
 $(AP) = \text{area under Precision vs Recall Curve}$
 1. For each detection (highest score to lowest score)
 1. If it matches some GT box with $\text{IoU} > 0.5$, mark it as positive and eliminate the GT
 2. Otherwise mark it as negative
 3. Plot a point on PR curve
 2. Average Precision (AP) = area under PR curve

How to get $AP = 1.0$: Hit all GT boxes with $\text{IoU} > 0.5$, and have no “false positive” detections ranked above any “true positives”



Evaluating Object Detectors: Mean Average Precision (mAP)

1. Run object detector on all test images (with NMS)
2. For each category, compute Average Precision
(AP) = area under Precision vs Recall Curve
 1. For each detection (highest score to lowest score)
 1. If it matches some GT box with $\text{IoU} > 0.5$, mark it as positive and eliminate the GT
 2. Otherwise mark it as negative
 3. Plot a point on PR curve
 2. Average Precision (AP) = area under PR curve
3. Mean Average Precision (mAP) = average of AP for each category

Flipz AP = 0.60
Hershey's AP = 0.85
Reese's AP = 0.81
mAP@0.5 = 0.75

Next Time: Semantic Segmentation



DeepRob

Lecture 12
Object Detection
University of Michigan and University of Minnesota

